
# เอกสารประกอบการสอน

## รายวิชา การเขียนโปรแกรมคอมพิวเตอร์แบบวิชวล

## Visual Programming

**กิตติคุณ บุญเกตุ**

**สาขาวิชาสถิติประยุกต์**

**คณะวิทยาศาสตร์ มหาวิทยาลัยราชภัฏบุรีรัมย์**

# คำนำ

เอกสารประกอบการสอนรายวิชาการเขียนโปรแกรมคอมพิวเตอร์แบบวิชวล รหัสวิชา 4132303 เล่มนี้ได้เรียบเรียงขึ้นเพื่อให้นักศึกษา ได้ใช้เป็นแหล่งเรียนรู้ด้วยตนเอง ซึ่งสาระสำคัญตรงกับคำอธิบายรายวิชาในหลักสูตรสาขาวิชาสถิติประยุกต์ ของมหาวิทยาลัยราชภัฏบุรีรัมย์ พร้อมทั้งมีตัวอย่างและแบบฝึกหัดประกอบอย่างครบถ้วนเพื่อเสริมการเรียนรู้ได้ดียิ่งขึ้นสำหรับการเรียนการสอน โดยแต่ละหัวข้อมีความจำเป็นในการใช้เวลาเรียนรู้ การทำความเข้าใจ และการปฏิบัติให้เกิดทักษะแตกต่างกัน

รายวิชานี้มุ่งเน้นให้ผู้เรียนมีความรู้ ความเข้าใจเกี่ยวกับหลักพื้นฐานของการเขียนโปรแกรมคอมพิวเตอร์แบบวิชวล ส่วนประกอบคุณลักษณะ และเหตุการณ์ การออกแบบสร้างฟอร์มและเมนู การประมวลผลฐานข้อมูล การทำโครงงานพัฒนาระบบงานประมวลผลสารสนเทศ โดยใช้โปรแกรมภาษาแบบวิชวลภาษาใดภาษาหนึ่ง

ผู้เขียนขอขอบคุณเจ้าของเอกสาร หนังสือ ตำรา บทความ และสื่อต่าง ๆ ที่ผู้เขียนยึดถือเป็นบรรทัดฐานในการเรียบเรียง ขอขอบคุณเจ้าหน้าที่สำนักวิทยบริการและเทคโนโลยีสารสนเทศที่ได้อำนวยความสะดวกในการสืบค้นข้อมูล และขอขอบคุณผู้เป็นกำลังใจทุกท่าน จนผลงานสำเร็จลุล่วงลงได้ด้วยดี

ผู้เขียนหวังอย่างยิ่งว่าเอกสารประกอบการสอนฉบับนี้ จะช่วยเอื้อประโยชน์ต่อผู้เรียน ผู้ที่ต้องการศึกษาเพิ่มเติมความรู้ รวมถึงผู้สนใจทั่วไปได้นำไปใช้ประโยชน์ในชีวิตประจำวัน หรืออาจมองไกลถึงการใช้เป็นพื้นฐานเพื่อก้าวสู่การประกอบอาชีพในอนาคตได้ตามที่ตั้งใจไว้

กิตติคุณ บุญเกตุ

ตุลาคม 2563

# Chapter 1
# เครื่องมือและการใช้งาน Visual Basic เบื้องต้น

## 1.1 ทำความรู้จัก Microsoft Visual Studio

Microsoft Visual Studio คือ ชุดพัฒนาโปรแกรม (Integrated Development Environment) ประกอบด้วยโปรแกรมหลายๆ ตัวที่ใช้ในการสร้างโปรแกรมสำเร็จรูปบนระบบปฏิบัติการวินโดวส์ หรือใช้สร้างเว็บโปรแกรม สร้างเว็บบริการ จัดการฐานข้อมูล และอื่นๆ อีกมากมาย

Visual Studio ได้รวบรวมเครื่องมือพัฒนาต่างๆ ที่ใช้ในการพัฒนาโปรแกรมตั้งแต่ หน้าจอที่ใช้พัฒนาโปรแกรม (Development interface) เครื่องมือในการตรวจหาจุดผิดในโปรแกรม (Debugging tool) ตัวช่วยอัตโนมัติในการเขียนโปรแกรม (Wizard tool) ตัวจัดการฐานข้อมูล (Database management) และส่วนประกอบอื่นๆที่จำเป็นในการพัฒนาโปรแกรม นำมาประกอบกันเป็นชุด เรียกว่า **I**ntegrated **D**evelopment **E**nvironment (ชุดพัฒนาโปรแกรม) หรือเรียกย่อๆ ว่า **IDE**

## 1.2 ความสามารถของ Visual Studio

เป็นชุดพัฒนาจากบริษัทไมโครซอฟท์ (Microsoft) ใช้ในการสร้างสิ่งต่างๆ ต่อไปนี้

- ✓ โปรแกรม (Program)
- ✓ เว็บไซต์ (Website)
- ✓ โปรแกรมบนเว็บ (Web application)
- ✓ บริการบนเว็บ (Web service)
- ✓ คลาวด์แอพ (Cloud apps) จัดการ และเผยแพร่คลาวด์แอพผ่านระบบคลาวด์ของบริษัทไมโครซอฟท์ที่มีชื่อว่า Azure
- ✓ แอพบนอุปกรณ์พกพา (Mobile apps)
- ✓ เกมส์ (Games)

## 1.3 เครื่องมือในชุด Visual Studio

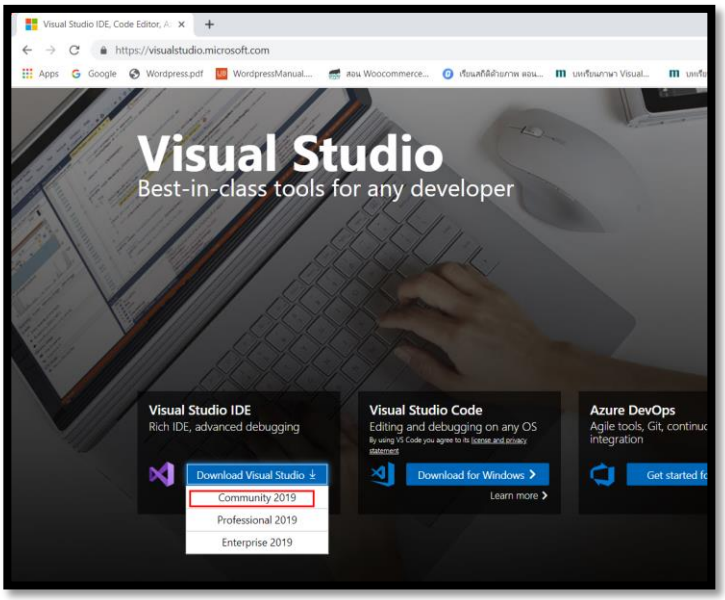
Visual Studio ได้รวมหลายๆ เครื่องมือที่ใช้ในการเขียนโปรแกรมภาษาต่างๆ และสามารถเขียนโปรแกรมจากภาษาใดก็ได้ตามที่เราถนัด โดยมีเครื่องมือดังต่อไปนี้

- ✓ Visual Basic
- ✓ Visual C++
- ✓ Visual C#
- ✓ Visual F#
- ✓ Python
- ✓ JavaScript

## 1.4 การติดตั้ง Visual Studio

ในการติดตั้ง Visual Studio ปัจจุบันนี้เป็น Version 2019 ซึ่งสามารถดาวน์โหลดได้ฟรี และติดตั้งตามขั้นตอนดังต่อไปนี้

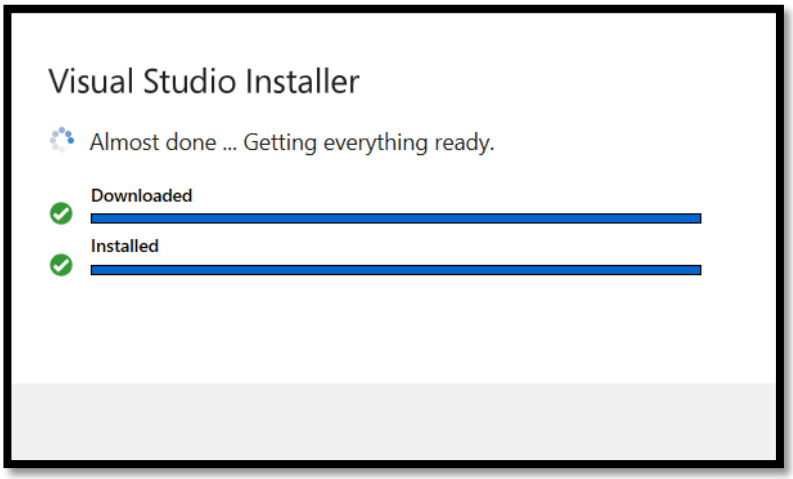
1.4.1 เริ่มต้นเข้าสู่เว็บไซต์ https://visualstudio.microsoft.com เมนู Visual Studio IDE เลือก Download Visual Studio -> Community 2019 ตามภาพที่ 1.1




**ภาพที่ 1.1**

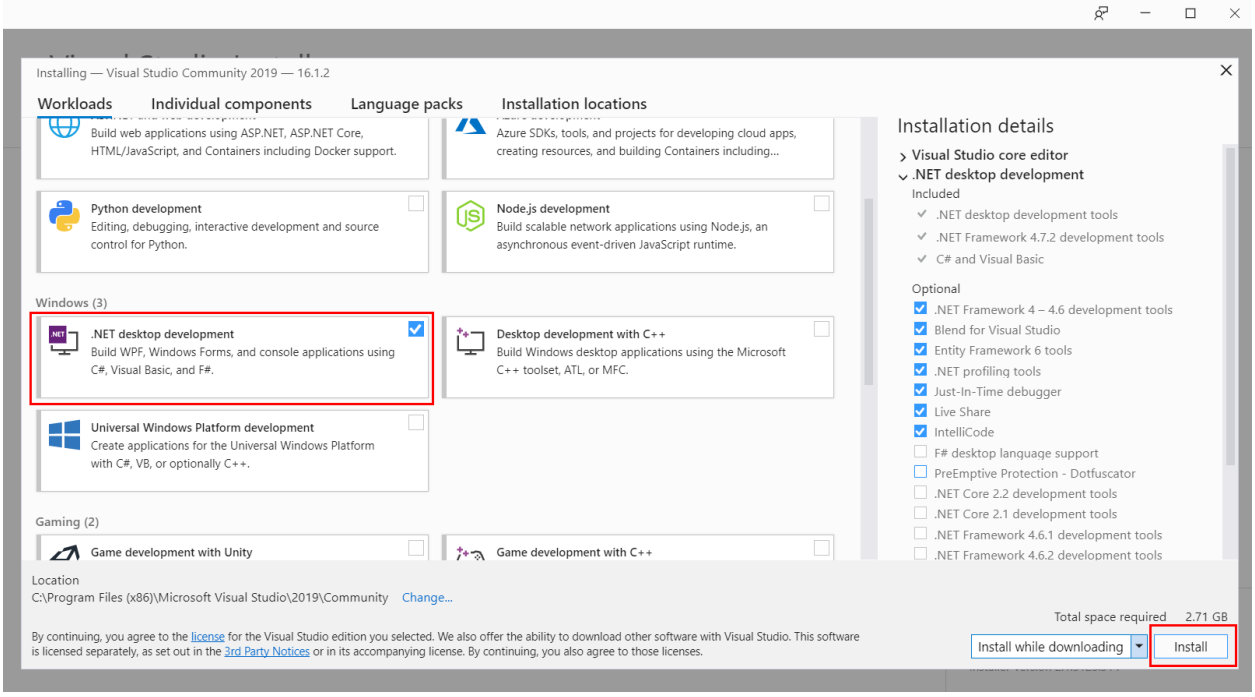
1.4.2 จะปรากฏแถบแสดงสถานะให้เลือกโฟลเดอร์ที่จะบันทึกโปรแกรมที่ดาวน์โหลดเก็บไว้ในเครื่อง

1.4.3 เปิดโฟลเดอร์ที่ทำการบันทึกโปรแกรมไว้ แล้วทำการ Run เพื่อทำการติดตั้ง จะปรากฏหน้าต่างแสดงสถานะการดาวน์โหลด Visual Studio ขั้นตอนนี้จะใช้เวลานานพอสมควร ตามภาพที่ 1.2



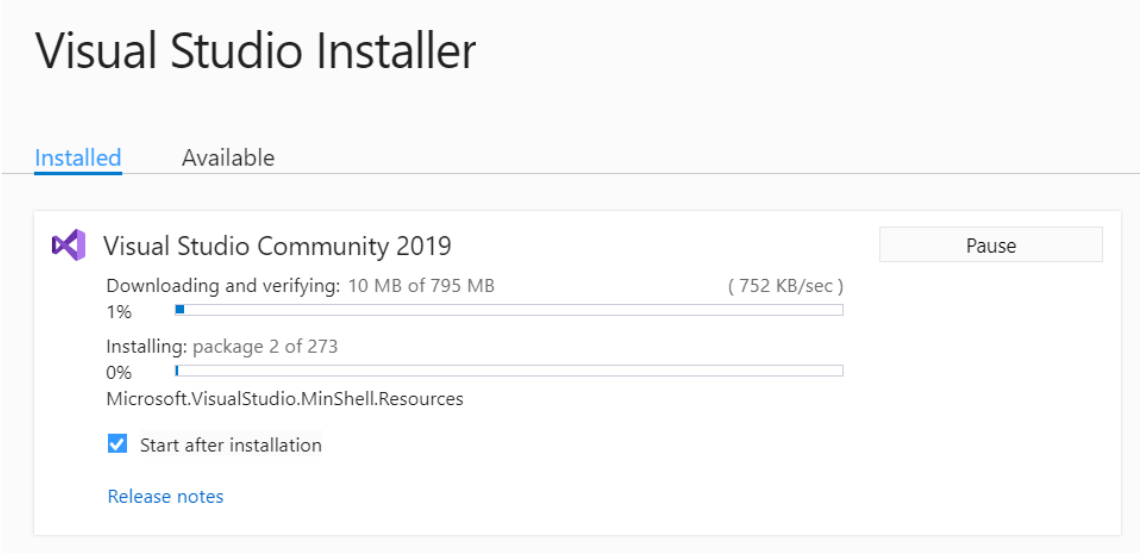

**ภาพที่ 1.2**

1.4.4 ขั้นตอนต่อไป เมื่อดาวน์โหลดเสร็จ จะมีหน้าต่าง installing ขึ้นมา ให้ทำการเลือกที่หัวข้อ .NET desktop development แล้วกดปุ่ม Install ตามภาพที่ 1.3



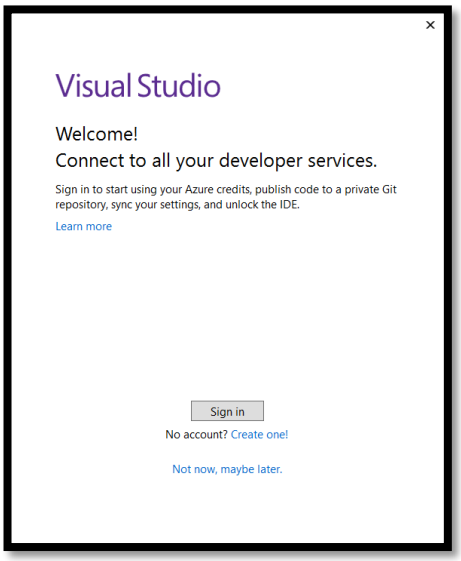

**ภาพที่ 1.3**

1.4.5 หลังจากกดปุ่ม install ทำการรอให้โปรแกรมดาวน์โหลดและติดตั้งอีกครั้งหนึ่ง ใช้เวลาพอสมควร ขึ้นอยู่กับความเร็วอินเทอร์เน็ต ตามภาพที่ 1.5




**ภาพที่ 1.5**

1.4.6 เมื่อติดตั้งโปรแกรมเสร็จเรียบร้อยแล้วทำการเปิดใช้งาน Visual Studio ครั้งแรก ให้ทำการลงทะเบียน หรือ Sign in เพื่อลงทะเบียนผู้ใช้ใหม่ (จะเหมือนกับการสมัครอีเมลล์ outlook) ทำการเลือกสภาพแวดล้อมที่ถนัด



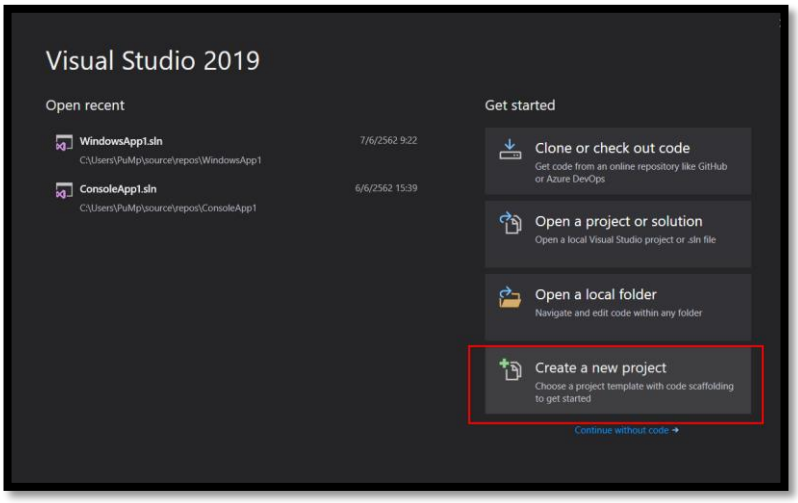

**ภาพที่ 1.6**

## 1.5 เริ่มต้นใช้งาน Visual Basic

เราจะเริ่มใช้งาน Visual Basic ซึ่งจะใช้คำย่อว่า VB โดยเราจะอธิบายถึงส่วนประกอบต่างๆ และกล่าวถึงหลักการเขียนโปรแกรมด้วย VB ที่จำเป็นต้องทราบ โดยประยุกต์ใช้สิ่งต่างๆ ดังนี้

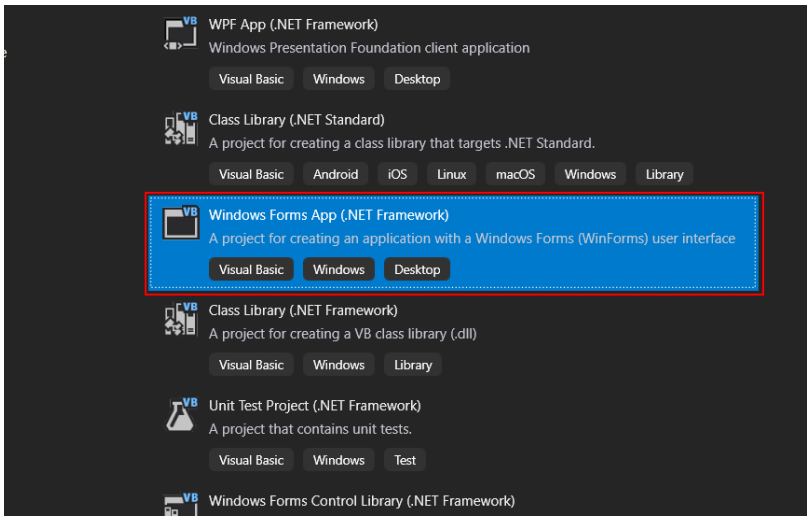
1.5.1 สร้างโปรเจ็กต์และรู้จักส่วนการทำงานใน Visual Studio

1) โปรเจกต์ (Project) คือ โครงการสร้างแอพพลิเคชันที่ประกอบด้วยไฟล์การทำงานรวมอยู่ในชุดเดียวกัน เริ่มต้นเราจะสร้างโปรเจ็กต์ใหม่ จากนั้นจะแสดงส่วนการทำงานต่างๆ ออกมามากมายให้ได้ศึกษา ดังนี้



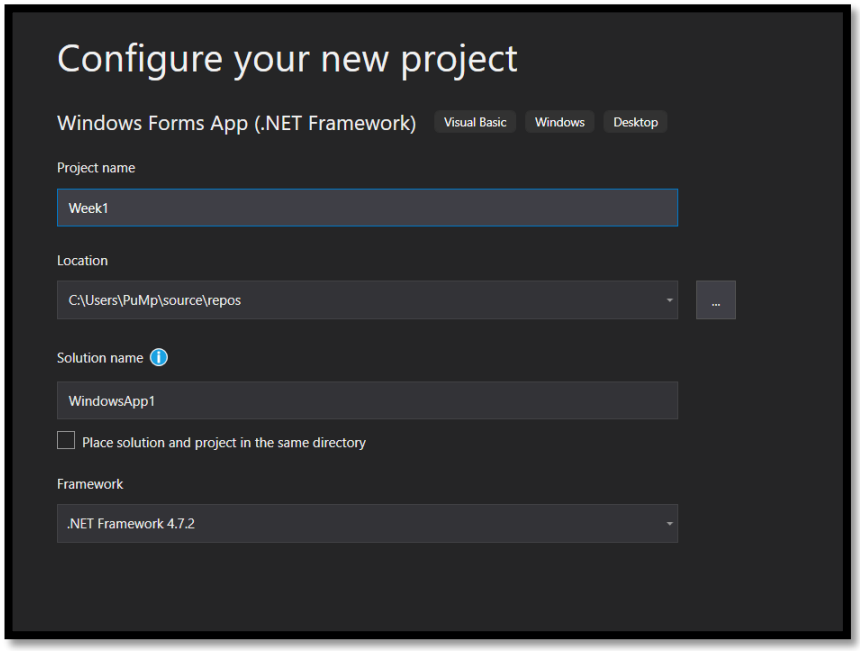

**ภาพที่ 1.7**

2) เลือกเทมเพลตต้นแบบ Windows Forms App (.Net Framework) แล้วกดปุ่ม Next ตามภาพที่ 1.8



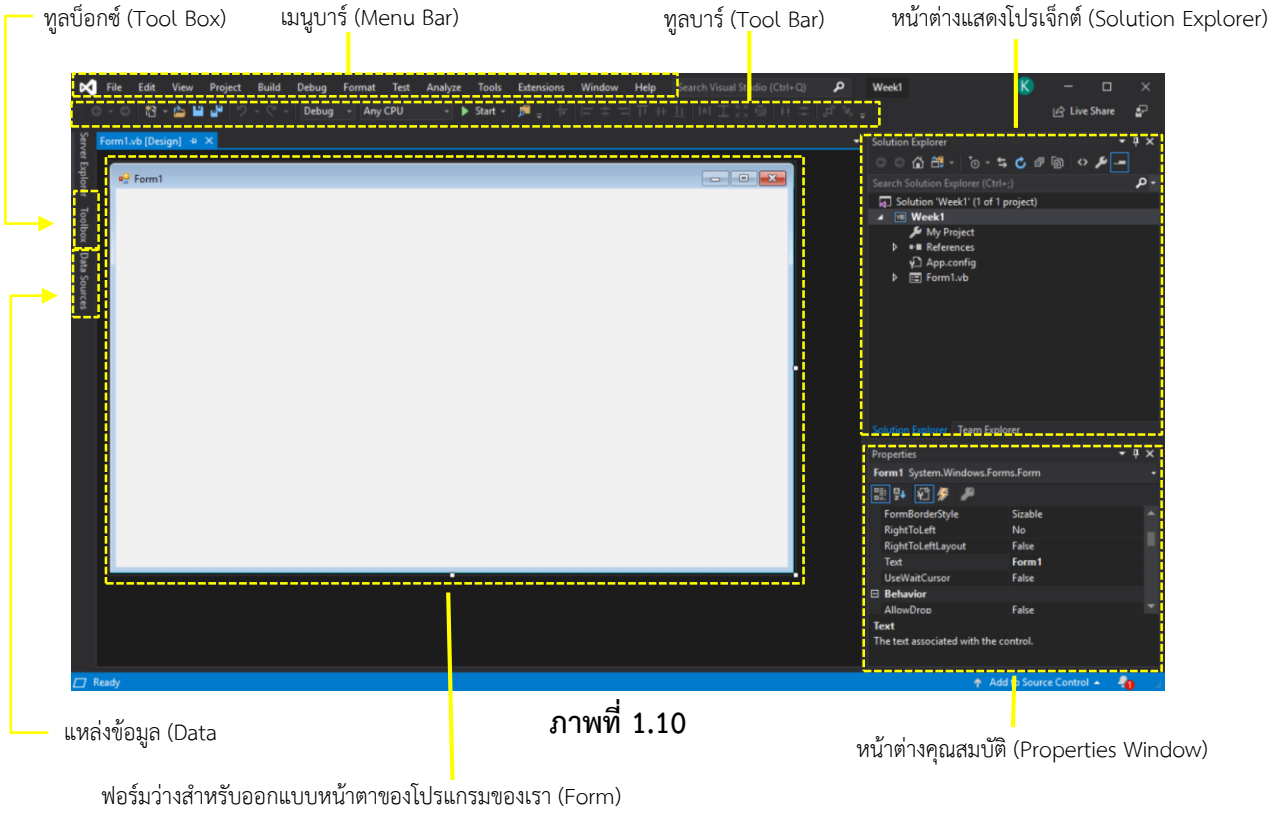

**ภาพที่ 1.8**

3) ทำการตั้งชื่อโปรเจ็กต์ เลือก Location ที่ต้องการบันทึก แล้วกดปุ่ม Create ภาพที่ 1.9




**ภาพที่ 1.9**

4) ปรากฏฟอร์มว่างสำหรับออกแบบหน้าจอโปรแกรมและแสดงส่วนการทำงานต่างๆ ภาพที่ 1.10 มีรายละเอียดดังนี้

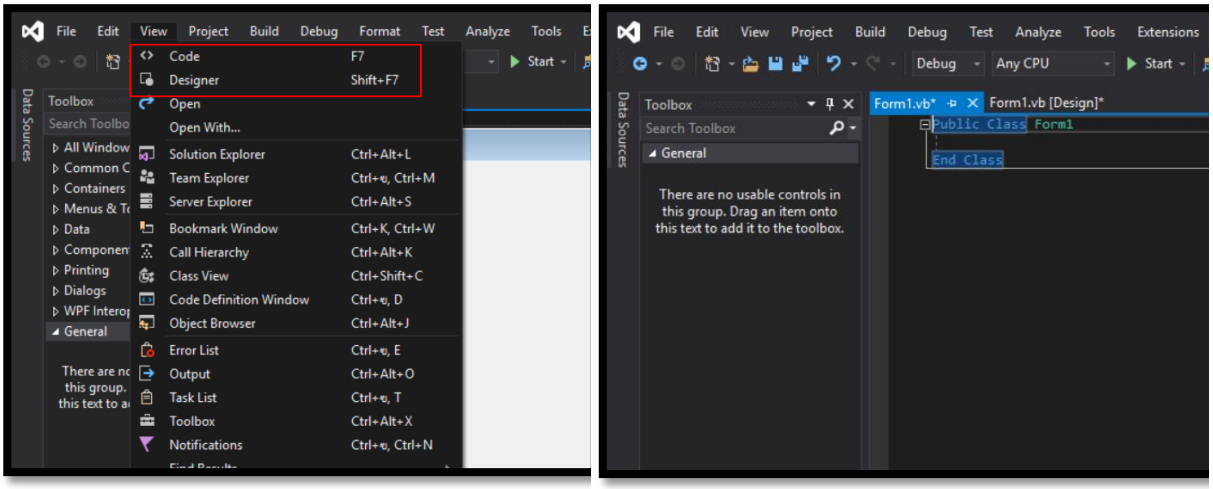



**ภาพที่ 1.10**

- เมนูบาร์ (Menu bar) เป็นแถบรวมคำสั่งทั้งหมดใน VB
- ทูลบาร์ (Toolbar) เก็บเฉพาะบางคำสั่ง แต่เป็นคำสั่งที่ใช้บ่อย
- ทูลบ็อกซ์ (Toolbox) เก็บคอนโทรลและคอมโพเนนต์ต่างๆ ที่เราสามารถเลือกไปวางลงบนฟอร์มได้ เพื่อออกแบบหน้าจอโปรแกรม (เรียกว่าส่วนติดต่อผู้ใช้ หรือ User Interface
- หน้าต่างแสดงโปรเจกต์ (Solution Explorer) เป็นหน้าต่างแสดงรายละเอียดของไฟล์และโมดูลที่มีในโปรเจ็กต์
- หน้าต่างคุณสมบัติ (Properties Window) เป็นหน้าต่างที่แสดงคุณสมบัติของคอนโทรลและคอมโพเนนต์ที่เลือกอยู่ในขณะนั้น ซึ่งเราสามารถปรับแต่งคุณสมบัติเพิ่มเติมได้ เช่น กำหนดข้อความ ปรับสี ปรับขนาด ขึ้นอยู่กับออบเจ็กต์ที่เลือก
- ฟอร์ม (Form) พื้นที่ที่ใช้ในการออกแบบหน้าจอโปรแกรม โดยให้เราลากคอนโทรลและคอมโพเนนต์มาจัดวางประกอบกันตามต้องการ
- แหล่งข้อมูล (Data Sources) สร้างการเชื่อมต่อและติดต่อฐานข้อมูล

5) หน้าต่าง Code Editor และหน้าต่าง Designer

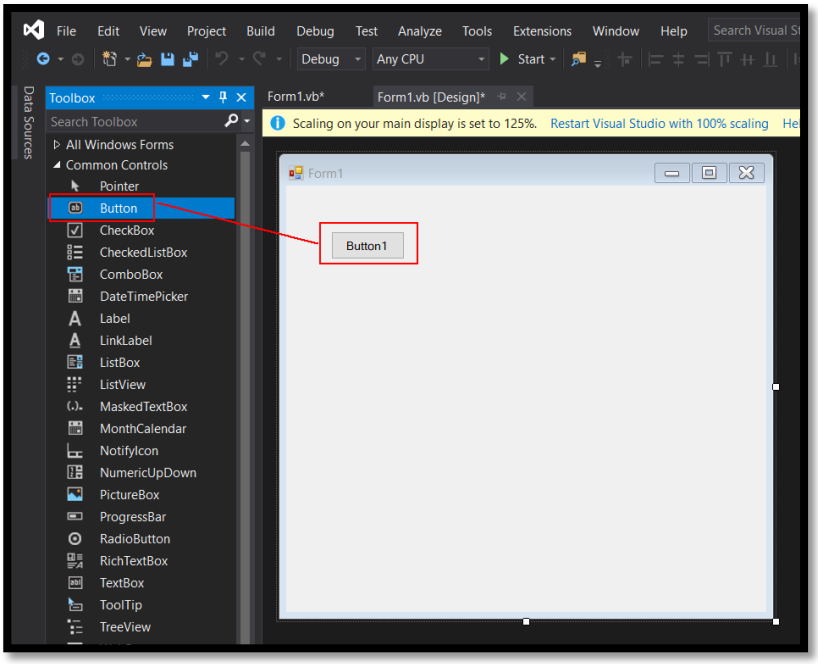
เราสามารถเรียกหน้าต่าง Code Editor ที่ให้ใส่คำสั่งเพื่อควบคุมการทำงานของโปรแกรม โดยการเลือกคำสั่ง View -> Code หรือกด F7 แต่ถ้าต้องการสลับไปทำงานบนหน้าต่าง Designer ให้เลือกคำสั่ง View -> Designer หรือกด Shift+F7 ตามภาพที่ 1.11




**ภาพที่ 1.11**

6) การจัดการคอนโทรลและการกำหนดอีเวนต์

6.1) การจัดการคอนโทรล การเพิ่มคอนโทรลลงบนฟอร์ม ให้ทำดังนี้ ภาพที่ 1.12 และเราสามารถกำหนดคุณสมบัติต่างๆได้ที่หน้าต่างคุณสมบัติ (Properties Window)




**ภาพที่ 1.12**

6.2) การกำหนดอีเวนต์

Code Editor เป็นหน้าต่างที่เราใส่คำสั่งของ VB ลงไป เพื่อตอบสนองต่ออีเวนต์ต่างๆ ที่เกิดขึ้น เราสามารถเปิด/ปิดหน้าต่างนี้ได้ โดยกดปุ่ม F7 หรือเลือกเมนู View->Code กรณีนี้ เราจะลองใช้คำสั่ง MessageBox.Show ในโปรแกรม แล้วทำการทดสอบโปรแกรม

```
Public Class Form1
    0 references
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Show1.Click
        MessageBox.Show("Hello Thailand")
    End Sub
End Class
```

**ภาพที่ 1.13**

7) การบันทึกโปรแกรม

โปรแกรมที่เราพัฒนาด้วย VB จะประกอบด้วยส่วนย่อยๆ เรียกว่า **โมดูล (Module)** ซึ่งโมดูลเหล่านี้ เมื่อนำมารวมกัน เราจะเรียกว่า **โปรเจ็กต์ (Project)** การบันทึกโปรแกรมที่เราสร้าง ก็คือ การบันทึกโปรเจ็กต์ ซึ่งจะมีอยู่ 3 วิธี คือ

- การบันทึกไฟล์ในโปรเจ็กต์ทั้งหมด ให้เราเลือก File -> Save All ซึ่งระบบจะสร้างโฟลเดอร์ย่อยสำหรับจัดเก็บโปรเจ็กต์และสร้างไฟล์โปรเจ็กต์ต่ามชื่อของโปรเจ็กต์ แลหากเราใช้คำสั่งนี้ในครั้งต่อไป ก็จะเป็นการบันทึกอัพเดตไฟล์ที่กำลังเปิดทำงานในโปรเจ็กต์ทั้งหมด
- การบันทึกโปรเจ็กต์ในชื่อเดิม ให้เราเลือก File -> Save [ชื่อของโปรเจ็กต์] ถ้ามีไฟล์ใหม่ที่เพิ่งสร้างแต่ยังไม่ถูกบันทึก VB จะให้เราตั้งชื่อไฟล์นั้นเพื่อทำการบันทึกด้วย
- การบันทึกโปรเจ็กต์ในชื่อใหม่ ให้เราเลือก File -> Save [ ] as จะเป็นการสร้างโปรเจ็กต์ชื่อใหม่ โดยโปรเจ็กต์ชื่อเดิมก็ยังคงอยู่เหมือนเดิม

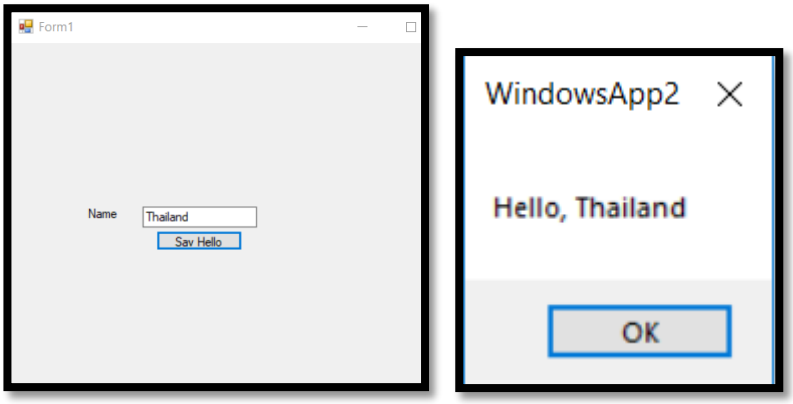
8) การทดสอบโปรแกรม

เราสามารถรันโปรแกรมเพื่อตรวจสอบการทำงาน โดยกดปุ่ม F5 หรือเลือกคำสั่ง Debug -> Start Debugging

หากต้องการยุติการทดสอบโปรแกรม ให้คลิกที่ปุ่มกากบาทปิดของโปรแกรม หรือกดปุ่ม <Ctrl+Alt+Bre-ak> หรือคำสั่ง Debug -> Stop Debugging

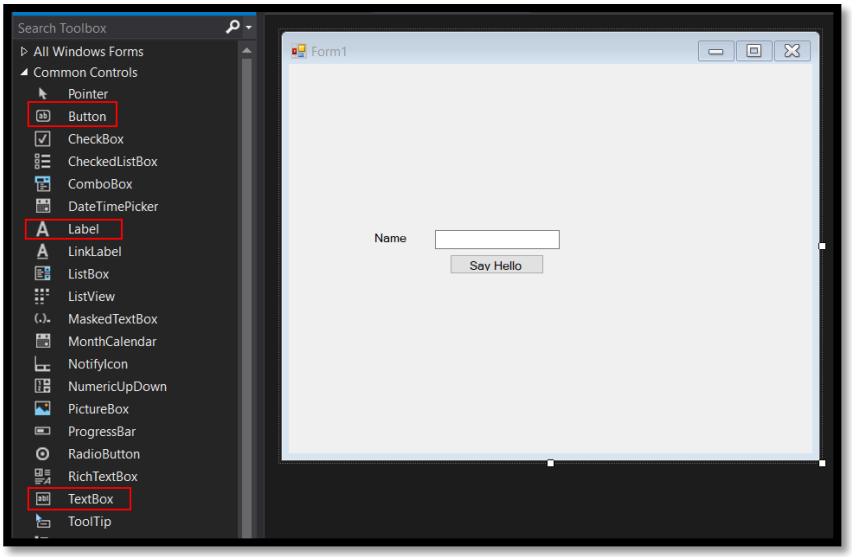
9) เริ่มต้นสร้างโปรแกรมแรก

สำหรับในโปรแกรมนี้ จะสร้างโปรแกรมที่รับข้อความจากผู้ใช้ เมื่อมีการคลิกเมาส์ปุ่ม Say Hello ก็จะแสดงข้อความที่ได้รับอยู่บนกล่องข้อความ ดังนี้




**ภาพที่ 1.14**

- เลือกคอนโทรลต่างๆ จากทูลบ็อกซ์ (ToolBox) มาวางบนฟอร์ม

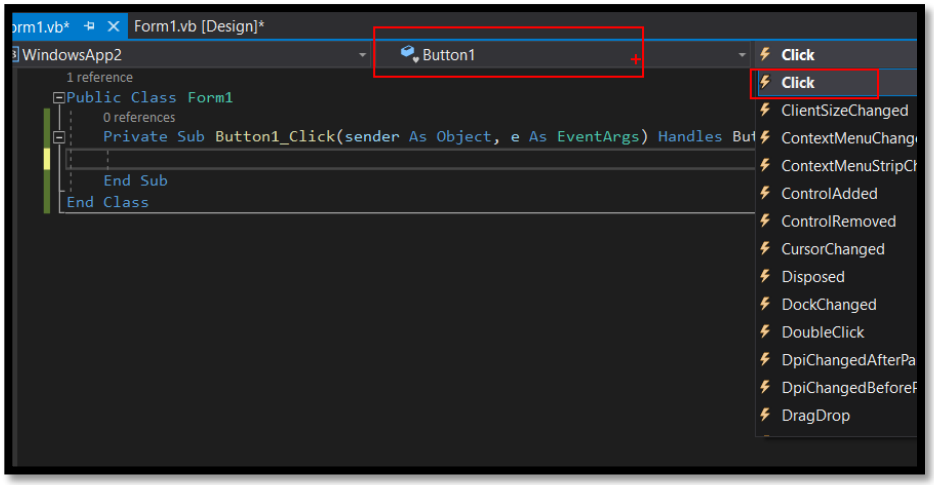



**ภาพที่ 1.15**

- กำหนดคุณสมบัติของคอนโทรลต่างๆ ที่หน้าต่าง Properties

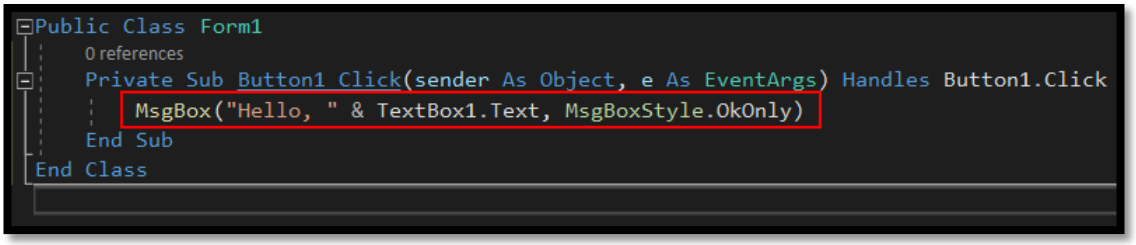
- เลือกเมนู View->Code เพื่อเปิดหน้าต่างเขียนโค้ดคำสั่ง (Code Editor)

- สร้างการตอบสนองที่ต้องการ ในตัวอย่างคือ กดปุ่ม Button1 ให้โชว์ข้อความ ดังนั้น หน้าต่างเขียน Code คำสั่ง ให้เราเลือก Button1 และเลือกอีเวนต์ Click ดังรูป




**ภาพที่ 1.16**

- ทำการเพิ่ม Code คำสั่งดังรูป




**ภาพที่ 1.17**

- คำสั่ง MsgBox จะแสดงกล่องข้อความ (Message Box) ที่มีข้อความแจ้งเตือน โดยเราจำกำหนดค่าไว้ 2 ค่า คั่นด้วยเครื่องหมา (,) คือข้อความที่แสดงและปุ่มกด

**คำแรก** “Hello, “ & TextBox1.Text แสดงคำว่า Hello (ตัวอักษรที่อยู่ในเครื่องหมาย (“ “) จะนับเป็นข้อความ ไม่ใช่คำสั่ง) จากนั้นต่อด้วยข้อความที่อยู่ใน TextBox1 ซึ่งจะเป็นข้อความที่เราพิมพ์ลงในคอนโทรล โดยทำการเชื่อมข้อความทั้งสองด้วยเครื่องหมาย &

**คำที่สอง** MsgBoxStyle.OkOnly เป็นการกำหนดปุ่มที่จะแสดงในกล่องข้อความ

- จากนั้นทำการรันโปรแกรม และทดสอบโปรแกรม

# Chapter 2

# การออกแบบหน้าจอและการใช้คอนโทรลพื้นฐาน

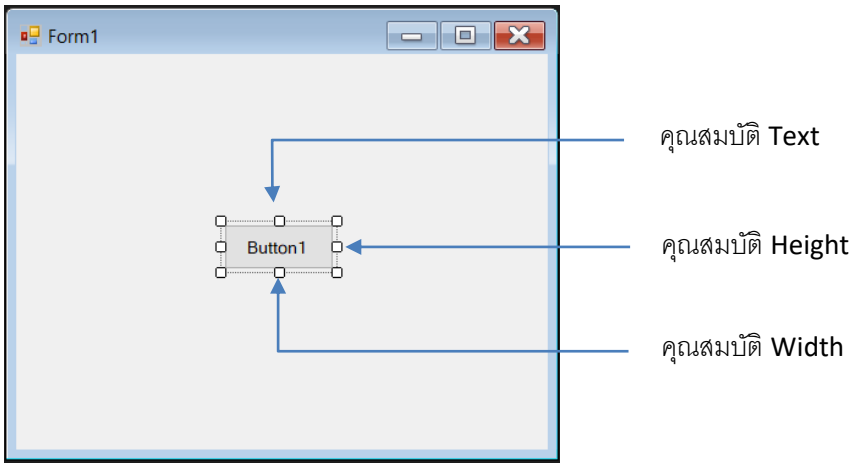
การสร้างโปรแกรมด้วย VB ในขั้นตอนแรกๆ จะเป็นการออกแบบหน้าจอที่ติดต่อกับผู้ใช้ โดยนำคอนโทรลที่มีอยู่มาออกแบบฟอร์มให้เหมาะสม จำเห็นได้ว่าทั้งฟอร์มและคอนโทรลนั้นเป็นเครื่องมือพื้นฐานที่สำคัญในการเขียนโปรแกรม VB สำหรับบนนี้จะกล่าวถึงหลักการพื้นฐานเกี่ยวกับฟอร์ม และคอนโทรลที่ควรทราบ เพื่อให้เข้าใจหลักการออกแบบหน้าจอมากยิ่งขึ้น

## 2.1 คุณสมบัติ เมธอด และอีเวนต์

ในการทำงานกับฟอร์มและคอนโทรล เราจะต้องทำความคุ้นเคยกับ 3 คำต่อไปนี้ คือ คุณสมบัติ (Properties), เมธอด (Method) และ อีเวนต์ (Event)

### 2.1.1 คุณสมบัติ (Properties)

เรากำหนดลักษณะต่างๆของ ฟอร์มและคอนโทรล เช่น ปุ่มคำสั่ง ชื่อ Button1 มีคุณสมบัติที่เรากำหนดได้




**ภาพที่ 2.1**

### 2.1.2 เมธอด (Method)

ความสามารถของ Object เป็นคำสั่งให้ฟอร์มและคอนโทรลทำงานตามที่เราต้องการ

2.1.3 อีเวนต์ (Event)

เป็นเหตุการณ์ที่เกิดขึ้นกับฟอร์ม หรือคอนโทรลที่เราสามารถใส่คำสั่งเพื่อตอบสนองได้ เช่น ถ้าเราต้องการตอบสนองต่ออีเวนต์ Click ของปุ่มคำสั่งชื่อ Button1 ให้เราดับเบิลคลิกปุ่มเพื่อเข้าสู่หน้าต่างการเขียนโค้ด และใส่คำสั่ง Button1.Text = “Click” จากนั้นเมื่อรันโปรแกรมให้คลิกปุ่มคำสั่งบนหน้าต่างโปรแกรม สังเกตเห็นคำว่า “Button1” บนปุ่มคำสั่ง จะเป็นเป็น “Click” ดังภาพที่ 2.2

```
    0 references
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Button1.Text = "Click"
    End Sub
End Class
```

**ภาพที่ 2.2**

## 2.2 เนมสเปซ (Namespaces)

หลักการที่สำคัญมากอย่างหนึ่งในการเขียนโปรแกรม VB ก็คือ เนมสเปซ ซึ่งจะช่วยให้การจัดการไลบรารีคลาส และออบเจ็กต์ต่างๆ ได้ง่ายขึ้น ทำให้ไม่เกิดความกำกวมเมื่ออ้างอิงถึงออบเจ็กต์ และระบบขอบเขตการใช้งานของออบเจ็กต์ รายละเอียดของเนมสเปซจะกล่าวในบทการเขียนโปรแกรมเชิงวัตถุ ส่วนบทนี้จะกล่าวถึงวิธีการอ้างอิงถึงเนมสเปซ

การอ้างอิงเนมสเปซ ทำให้เราเรียกใช้คลาสได้ง่าย เช่น คลาส System.WinForms.Button เป็นการเรียกชื่อแบบเต็มๆ ซึ่งยาวและยุ่งยาก แต่ถ้าเราอ้างอิงด้วยคำสั่ง import ดังตัวอย่าง

```
Imports System.WinForms
```

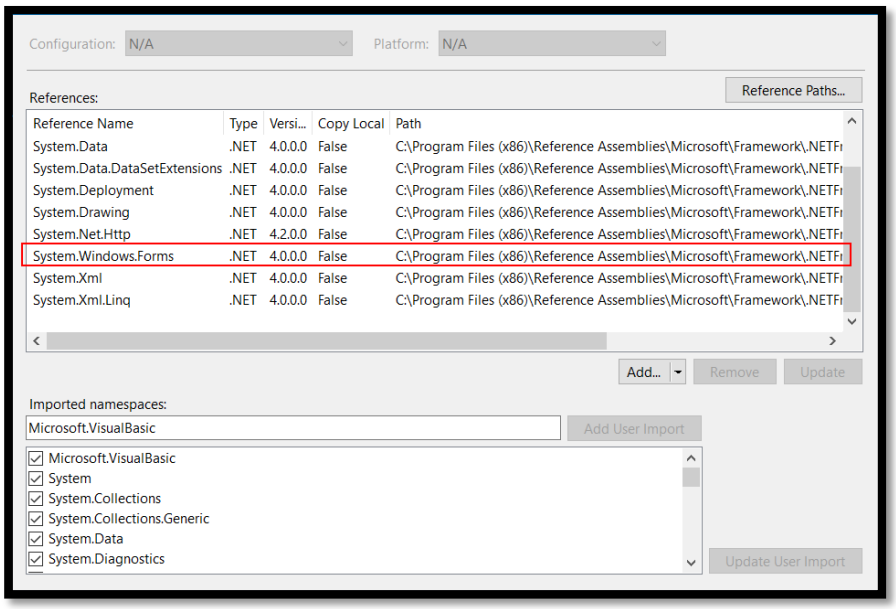
ทำให้เราสามารถเรียกใช้คลาสภายใน WinForms ทั้งหมดได้ในทันที เช่น เรียกใช้คลาส Button แทนการใช้คำว่า System.WinForms.Button ซึ่งจะสั้นกว่าและสะดวกกว่ามาก

## 2.3 การใช้งานฟอร์ม (Windows Forms)

2.3.1 ฟอร์ม หรือที่เรียกว่า WinForms เป็นเครื่องมือที่ใช้บ่อยมากในการสร้างโปรแกรมด้วย VB โดยผู้ใช้จะติดต่อทำงานผ่านทางคอนโทรลต่างๆ ที่เราวางบนฟอร์ม

ฟอร์มและคอนโทรลต่างๆ ที่มีใน VB นั้น เป็นคลาสที่มีอยู่ในเนมสเปซชื่อ System.Windows.Forms ซึ่งจะมีคลาสต่างๆ ที่ช่วยเราในการสร้างแอพพลิเคชันที่รันบนวินโดวส์ (สำหรับความหมายของคลาสและออบ

เจ็กต์ เราจะกล่าวถึงรายละเอียดเพิ่มเติม ในบทเรื่องการเขียนโปรแกรมเชิงวัตถุ) โดยเนมสเปซ System.Windows.Forms จะมีใช้ในโปรเจ็กต์เราอยู่แล้วตั้งแต่แรก โดยสามารถเข้าไปดูได้ที่หน้าต่าง Solution คลิกขวาที่ชื่อโปรเจคแล้วเลือก Properties จากนั้นไปที่เมนู References ดังภาพที่ 2.3




**ภาพที่ 2.3**

2.3.2 คุณสมบัติที่สำคัญของฟอร์ม

ฟอร์มมีคุณสมบัติต่างๆ มากมาย แต่ที่เราจำเป็นต้องรู้จัก มีดังต่อไปนี้

| ชื่อคุณสมบัติ | คำอธิบาย |
|---|---|
| Name * | กำหนดชื่อของฟอร์มที่เราใช้อ้างอิงถึงในโปรแกรม |
| FormBorderStyle | กำหนดลักษณะการเปลี่ยนขนาดขอบของฟอร์ม |
| MinimizeBox หรือ MaximizeBox | กำหนดว่าให้ขยายหรือย่อฟอร์ม |
| ControlBox | กำหนดให้ฟอร์มมีเมนูระบบทางมุมซ้ายบนสุดด้วยหรือไม่ |
| Text * | กำหนดข้อความที่แสดงบนไตเติลบาร์ของฟอร์ม |
| Icon | กำหนดรูปไอคอนของฟอร์มเมื่อย่อ (Minimize) ฟอร์ม |
| Size * | กำหนด Height : ความสูง และ Width : ความกว้างของฟอร์ม |
| Location * | กำหนดตำแหน่งของฟอร์มโดยคิดจากตำแหน่งบนซ้ายของหน้าจอ |

| ชื่อคุณสมบัติ | คำอธิบาย |
|---|---|
| WindowState | กำหนดว่าเมื่อเริ่มต้นเรียกฟอร์ม จะให้อยู่ในแบบขยาย (Maximize), แบบย่อ (Minimize) หรือแบบธรรมดา |
| Enabled * | กำหนดให้ฟอร์มสามารถตอบสนองต่ออีเวนต์ได้หรือไม่ |
| TopMost | กำหนดว่าฟอร์มนี้จะแสดงอยู่เหนือฟอร์มอื่นๆ เสมอหรือไม่ |

หมายเหตุ * เป็นคุณสมบัติที่ใช้กับคอนโทรลอื่นได้ โดยมีความหมายใกล้เคียงกัน

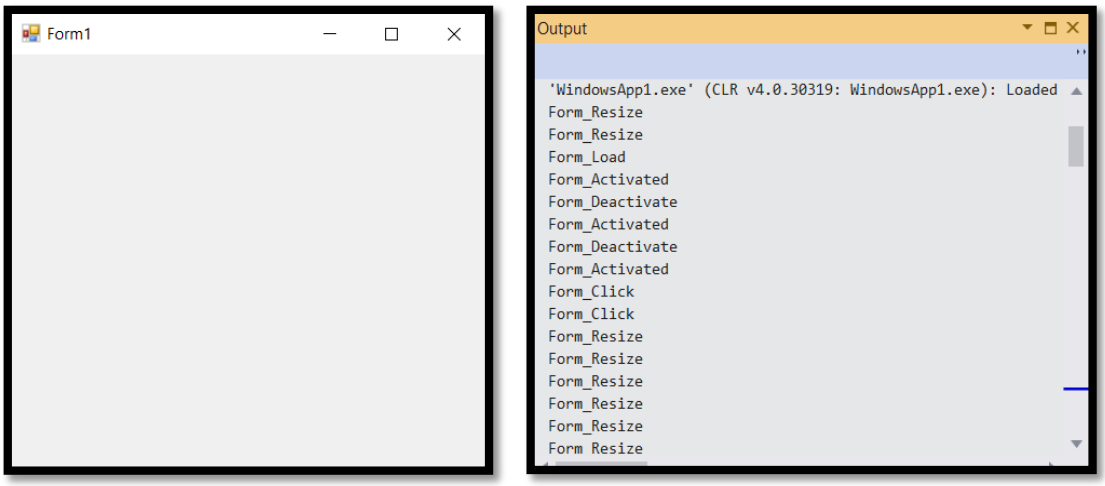
2.3.3 เมธอดและอีเวนต์ที่สำคัญของฟอร์ม

ฟอร์มมีเมธอดและอีเวนต์ที่เราจำเป็นต้องรู้จัก ดังตารางต่อไปนี้

| ชื่อเมธอด, อีเวนต์ | คำอธิบาย |
|---|---|
| Load | เป็นอีเวนต์ที่เกิดเมื่อเรียกฟอร์มขึ้นมาครั้งแรก |
| Resize | เป็นอีเวนต์ที่เกิดเมื่อมีการเปลี่ยนขนาดของฟอร์ม |
| Activated | เป็นอีเวนต์ที่เกิดขึ้นเมื่อฟอร์มนั้นเป็นฟอร์มที่เรากำลังทำงานด้วย |
| Deactivate | เป็นอีเวนต์ที่เกิดเมื่อฟอร์มอื่นๆ กลายเป็นฟอร์มที่แอคทีฟแทน |
| Click | เป็นอีเวนต์ที่เกิดเมื่อมีการใช้เมาส์คลิกบนฟอร์ม |
| DoubleClick | เป็นอีเวนต์ที่เกิดเมื่อมีการใช้เมาส์ดับเบิ้ลคลิกบนฟอร์ม |
| Show | เป็นเมธอดที่ใช้แสดงฟอร์มขึ้นมา |
| Hide | เป็นเมธอดที่ใช้ซ่อนฟอร์มไว้ |
| Close | เป็นเมธอดที่ใช้ปิดฟอร์ม |
| Activate | เป็นเมธอดที่ใช้ทำให้ฟอร์มถูกเลือกใช้งานในขณะนั้น |

2.3.4 ตัวอย่างโปรแกรมแสดงการทำงานของฟอร์ม

ตัวอย่างนี้เป็นเป็นโปรแกรมที่ช่วยให้เราเข้าใจสิ่งต่างๆ เกี่ยวกับฟอร์มมากยิ่งขึ้น ซึ่งเป็นการสร้างฟอร์มหนึ่งฟอร์ม และแสดงอีเวนต์ต่างๆ ทีเกิดขึ้นบนฟอร์มว่าเกิดขึ้นในตอนใด ดูผลลัพธ์ได้จากหน้าต่าง Output โดยโปรแกรมจะเรียกเมธอด WriteLine ของออบเจ็กต์ Console คำสั่ง Console.WriteLine เพื่อพิมพ์ชื่ออีเวนต์ที่เกิดขึ้นบนฟอร์มออกมา โดยมีการทำงานดังนี้

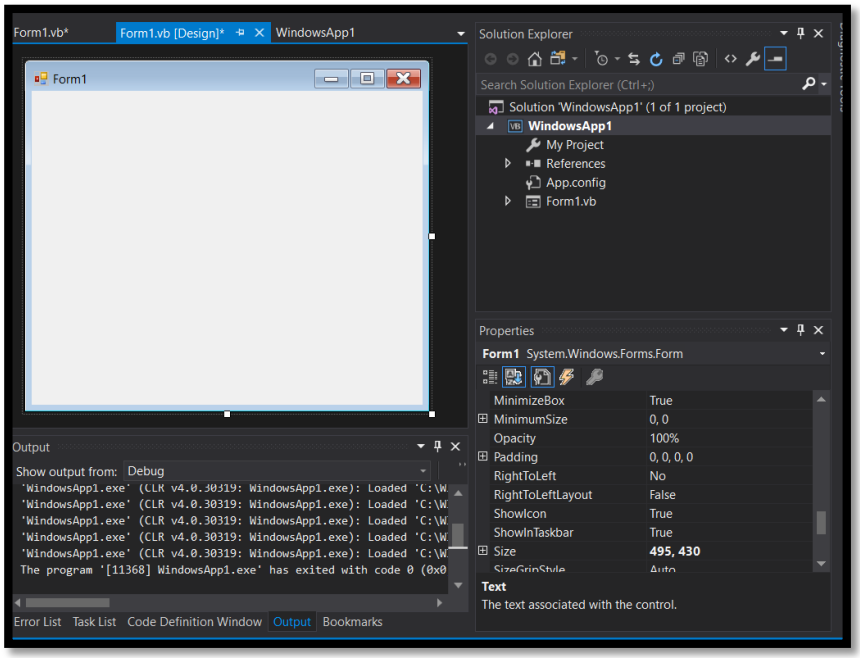



**ภาพที่ 2.4**

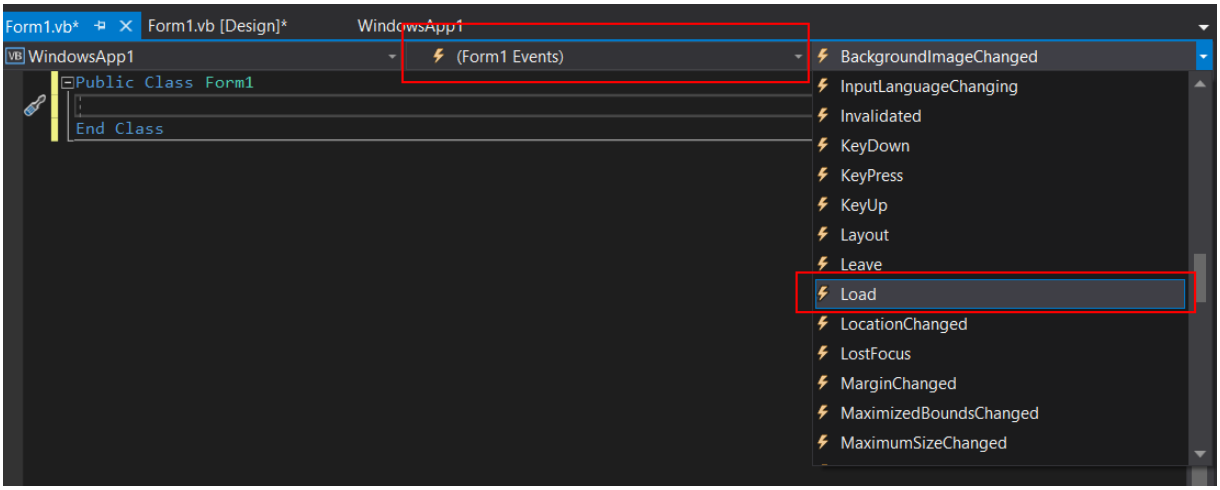
**ขั้นตอนการสร้างโปรเจ็กต์**

1) สร้างโปรเจ็กต์ใหม่ชนิด Windows Forms App

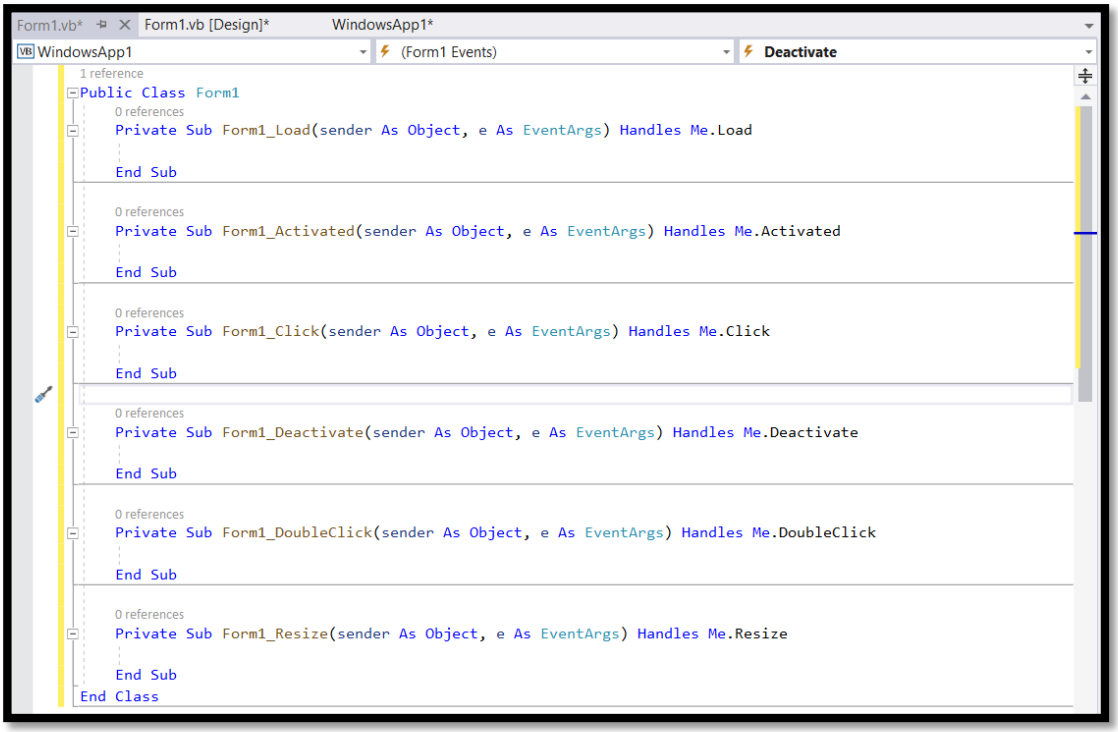
2) เลือก View -> Code หรือ กด F7 เพื่อเปิดหน้าต่างโค้ดขึ้นมา จากนั้นให้เลือกอีเวนต์ต่างๆ ของฟอร์มตามคำสั่งในโปรแกรม โดยเลือกจาก (Form1 Events) และเลือกอีเวนต์ของฟอร์มที่ต้องการตอบสนอง ในที่นี้เลือก 6 อีเวนต์ คือ Load, Activated, Click, Deactivate, DoubleClick, Resize




**ภาพที่ 2.5** สร้างโปรเจ็กต์ใหม่




**ภาพที่ 2.6** เลือก (Form1 Events) และเลือกอีเวนต์ตอบสนอง 6 ตัว



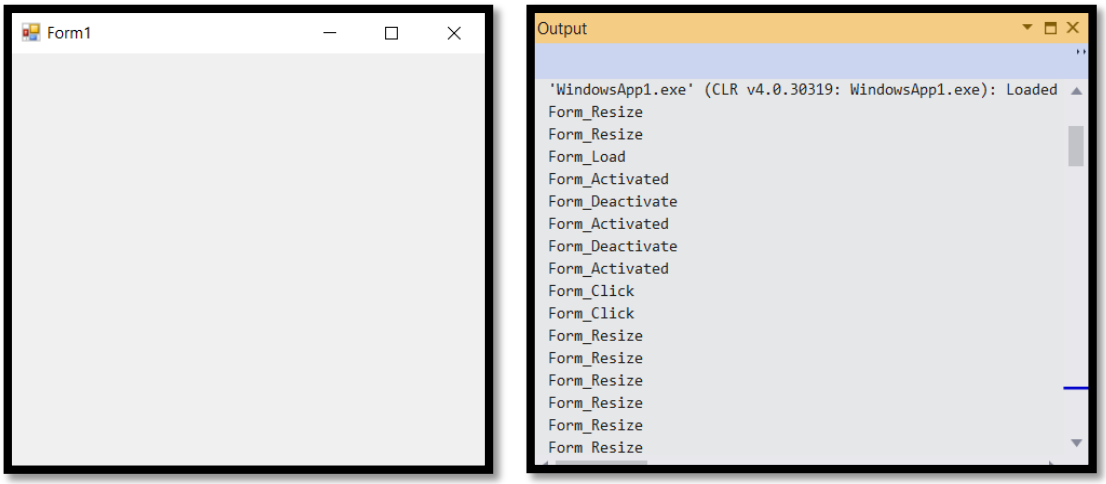

**ภาพที่ 2.7** โปรแกรมย่อยหลังจากที่เลือกอีเวนต์

3) จากนั้นในหน้าต่างโค้ดโปรแกรม ให้เราใส่คำสั่งโปรแกรมต่อไปนี้ลงไปในอีเวนต์ต่างๆ ดังนี้

```
Public Class Form1
    0 references
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles Me.Load
        Console.WriteLine("Form_Load")      'ตอบสนองต่ออีเวนต์ Load
    End Sub
    0 references
    Private Sub Form1_Activated(sender As Object, e As EventArgs) Handles Me.Activated
        Console.WriteLine("Form_Activated")    'ตอบสนองต่ออีเวนต์ Activated
    End Sub
    0 references
    Private Sub Form1_Click(sender As Object, e As EventArgs) Handles Me.Click
        Console.WriteLine("Form_Click")        'ตอบสนองต่ออีเวนต์ Click
    End Sub
    0 references
    Private Sub Form1_Deactivate(sender As Object, e As EventArgs) Handles Me.Deactivate
        Console.WriteLine("Form_Deactivate")     'ตอบสนองต่ออีเวนต์ Deactivate
    End Sub
    0 references
    Private Sub Form1_DoubleClick(sender As Object, e As EventArgs) Handles Me.DoubleClick
        Console.WriteLine("Form_DoubleCLick")     'ตอบสนองต่ออีเวนต์ DoubleClick
    End Sub
    0 references
    Private Sub Form1_Resize(sender As Object, e As EventArgs) Handles Me.Resize
        Console.WriteLine("Form_Resize")        'ตอบสนองต่ออีเวนต์ Resize
    End Sub
End Class
```

**ภาพที่ 2.8**

4) บันทึกโปรเจ็กต์และรันทดสอบโปรแกรม โดยเลือก Debug และคลิก Start โดยผลลัพธ์การรันโปรแกรมจะปรากฏชื่อ Form1 ขึ้นมา และเมื่อมีอีเวนต์เกิดขึ้น โปรแกรมจะพิมพ์อีเวนต์ที่เกิดขึ้นบนหน้าต่าง Output ดังรูป




**ภาพที่ 2.9**

**<u>เพิ่มเติม</u>** คำสั่ง Console.WriteLine จะมีประโยชน์มากในการแสดงค่าข้อมูลในโปรแกรมของเราออกมาทางหน้าต่าง Output สามารถเรียกขึ้นมาแสดงได้โดยไปที่เมนู View-> Output หรือกดปุ่ม <Ctrl+Alt+O>

2.3.5 คอนโทรลและคอมโพเนนต์พื้นฐาน

ทูลบ็อกซ์ ประกอบด้วยเครื่องมือที่ใช้นำมาวางบนฟอร์มเพื่อออกแบบส่วนติดต่อผู้ใช้งาน มีเครื่องมือ 2 ประเภท คือ **คอนโทรล** ออบเจ็กต์ที่วางบนฟอร์มที่ผู้ใช้สามารถมองเห็น เช่น ปุ่ม กล่องข้อความ เป็นต้น และ **คอมโพเนนต์** ออบเจ็กต์ที่วางบนฟอร์มแต่ผู้ใช้มองไม่เห็น เช่น การนับเวลา เป็นต้น

| คอนโทรล | คำอธิบาย |
|---|---|
| Button | ให้ผู้ใช้คลิกสั่งงานมายังโปรแกรม |
| Label | แสดงข้อมูลให้ผู้ใช้เห็นเพียงอย่างเดียว |
| Textbox | รับข้อมูลจากผู้ใช้ที่ป้อนเข้ามาในโปรแกรมและแสดงผลข้อมูล |
| CheckBox | ให้ผู้ใช้เลือกได้มากกว่า 1 ตัวเลือก |
| RadioButton | ให้ผู้ใช้เลือกได้ตัวใดตัวหนึ่งจากกลุ่มตัวเลือกทั้งหมด |
| ListBox | ให้ผู้ใช้เลือกโดยการเลื่อนดูรายการที่ต้องการได้ |
| ComboBox | ให้ผู้ใช้เลือกคล้ายกับลิสต์บ็อกซ์ แต่สามารถพิมพ์ข้อมูลเข้าไปได้ด้วย |
| CheckedListBox | ให้ผู้ใช้เลือกรายการได้โดยเช็กที่หน้ารายการนั้น |
| GroupBox | รวมคอนโทรลให้อยู่ในกลุ่มเดียวกัน |
| PictureBox | ใส่ภาพประกอบและตกแต่งให้สื่อความหมายและสวยงาม |
| ScrollBar | เลื่อนดูข้อมูลทั้งหมดที่ไม่สามารถแสดงในคราวเดียวบนพื้นที่ที่จำกัด |

| คอมโพเนนต์ | คำอธิบาย |
|---|---|
| Timer | ใช้นับเวลาตามช่วงเวลาที่กำหนดเพื่อควบคอมการทำงานบางอย่าง |

2.3.5.1 คอนโทรลปุ่มคำสั่ง (Button)

ปุ่มคำสั่งมีหน้าที่หลักคือ ตอบสนองต่อการคลิกเมาส์ของผู้ใช้งานที่สั่งงานมายังโปรแกรมว่าต้องการให้โปรแกรมทำอะไรต่อ

ตัวอย่างโปรแกรมที่ใช้ปุ่มคำสั่ง

ในตัวอย่างนี้ จะแสดงการใช้งานปุ่มคำสั่ง ถ้าเราไม่ใส่ข้อความในเท็กซ์บ็อกซ์ ปุ่มคำสั่งจะไม่ตอบสนองการทำงาน แต่ถ้าเราใส่ข้อความในเท็กซ์บ็อกซ์ ปุ่มคำสั่งจะตอบสนองต่อการคลิกเมาส์ โดยแสดงชื่อที่พิมพ์ในเท็กซ์บ็อกซ์บนกล่องข้อความ

1) สร้างโปรเจคใหม่ชนิด Windows Forms App จากนั้นให้วางคอนโทรล และเปลี่ยนชื่อปุ่มกับลาเบล ดังภาพที่ 2.10


**ภาพที่ 2.10**

2) สังเกตได้ว่าที่ปุ่มมีรูปภาพไอคอนเพิ่มขึ้นมา โดยโปรแกรม Visual Studio มีคลังภาพให้นักพัฒนาโปรแกรมสามารถดาวน์โหลดมาใช้ประกอบในโปรแกรมได้ฟรี เรียกว่า Visual Studio Image Library ในตัวอย่างเราจะใช้ไฟล์ภาพ UserProfile_16x.png มาตกแต่งบนปุ่ม มีขั้นตอนดังนี้

- เปิดเว็บ Google แล้ว Search คำว่า Microsoft Visual Studio Image Library
- เข้าเว็บ https://www.microsoft.com/en-us/download/details.aspx?id=35825
- คลิกที่ Download
- เลือก VS2017 Image Library.zip จากนั้นคลิกปุ่ม Next
- ทำการ Save บันทึกไว้ในเครื่องคอมพิวเตอร์
- จะได้ไฟล์ VS2017 Image Library.zip ที่เก็บภาพไอคอนและกราฟฟิกจำนวนมาก

- จากนั้นแตกไฟล์ .zip ออกมา แล้วหาโฟลเดอร์ UserProfile

- ก็อปปี้ไฟล์ UserProfile_16x.png มาใส่ในโฟลเดอร์โปรเจ็กต์ของเรา

- คลิกเลือกคอนโทรลปุ่มที่เราต้องการวางภาพประกอบ

- ที่หน้าต่าง Properties ให้คลิกกำหนดคุณสมบัติ Image ที่  เพื่อใส่ภาพปุ่ม

- จะปรากฏหน้าต่าง Select Resource ที่ Project resource file ให้คลิก Import เพื่อเข้าไปเลือกไฟล์ภาพที่ต้องการนำมาวางบนปุ่ม

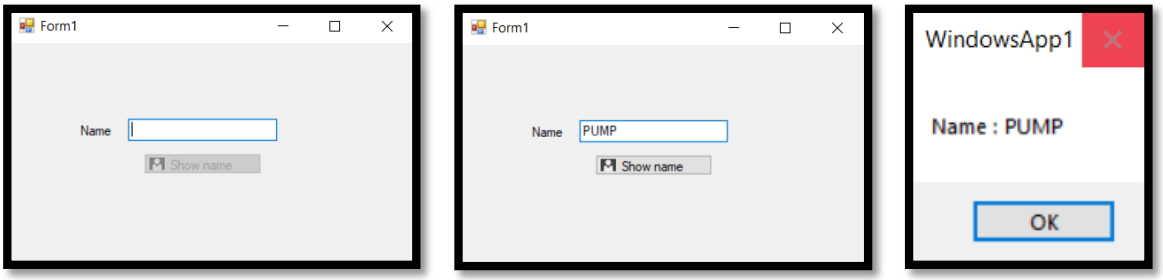
- ปรากฏหน้าต่าง Open ให้เลือกไฟล์ภาพที่จัดเตรียมเพื่อนำมาวางบนปุ่มและคลิก Open

- กลับมาที่หน้าต่าง Select Resource ที่ Project resource file ให้เลือก UserProfil_16x และคลิกปุ่ม OK จากนั้นภาพจะถูกวางบนปุ่ม ทำการจัดตำแหน่งให้เรียบร้อยที่หน้าต่าง Properties กำหนดคุณสมบัติที่ ImageAlign

3) ดับเบิลคลิกบนปุ่มเพื่อเข้าสู่ Code Editor จากนั้นสร้างโค้ดอีเวนต์ TextBox1_TextChanged แล้วให้ใส่คำสั่งภายในอีเวนต์ต่างๆ ดังนี้

```
1 reference
Public Class Form1
    0 references
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        'สร้างกล่องข้อความแสดงข้อความของ TextBox1
        MsgBox("Name : " & TextBox1.Text, MsgBoxStyle.OkOnly)
    End Sub
    0 references
    Private Sub TextBox1_TextChanged(sender As Object, e As EventArgs) Handles TextBox1.TextChanged
        'ตรวจสอบข้อความใน TextBox1 เป็นข้อความว่างหรือไม่
        If TextBox1.Text <> "" Then
            Button1.Enabled = True      'ถ้าไม่ว่าง ทำให้ปุ่มคำสั่งสามารถทำงานได้
        Else
            Button1.Enabled = False     'ถ้าว่าง ทำให้ปุ่มคำสั่งทำงานไม่ได้
        End If
    End Sub
End Class
```

**ภาพที่ 2.10**




**ภาพที่ 2.11**

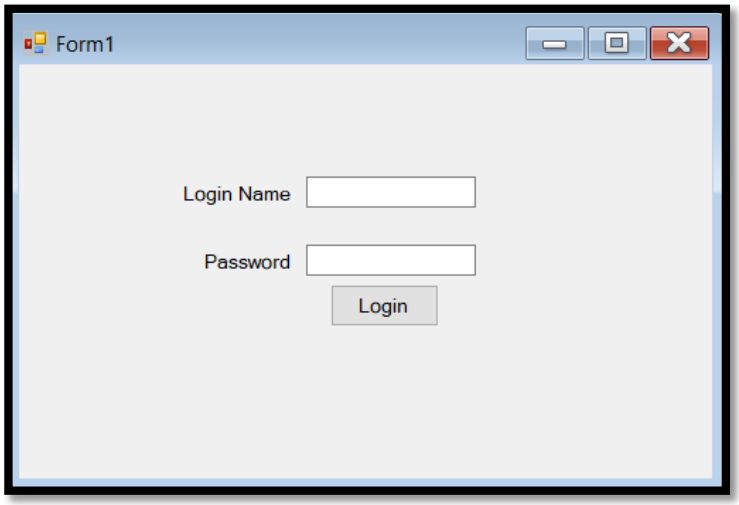
2.3.5.2 คอนโทรลเท็กซ์บ็อกซ์ (TextBox)

เป็นคอนโทรลที่เราใช้บ่อยมากในการรับข้อมูลจากผู้ใช้ที่ป้อนเข้ามาในโปรแกรม รวมทั้งสามารถแสดงผลและให้ผู้ใช้แก้ไขข้อมูลได้ด้วย การปรับแต่งจะอยู่ในส่วยของ Properties

<u>ตัวอย่างโปรแกรมแสดงการใช้งานเลเบล เท็กซ์บ็อกซ์ และปุ่มกด</u>

<u>ขั้นตอนการสร้างโปรแกรม</u>

1) สร้างโปรเจคใหม่เป็นชนิด Windows Forms App จากนั้นวางคอนโทรลต่างๆ บนฟอร์ม และกำหนดคุณสมบัติ ดังนี้



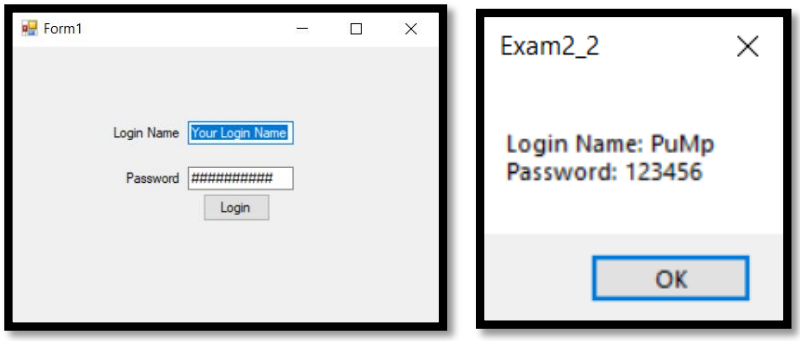

**ภาพที่ 2.12**

2) ดับเบิ้ลคลิกที่ปุ่ม Login เพื่อเข้าสู่หน้า Code Editor ให้ใส่คำสั่งใน Event Button1_Click จากนั้นดับเบิลคลิกที่ว่างบนฟอร์มสร้างโค้ดอีเวนต์ Form1_Load และใส่โค้ดการทำงานดังนี้

```
Public Class Form1
    0 references
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        'เมื่อคลิกปุ่มกดจะปรากฏกล่องข้อความแสดงชื่อล็อกอินและรหัสผ่านที่ใส่เข้าไป
        'vbnewline เป็นคำสั่งที่ใช้ขึ้นบรรทัดใหม่ใช้คู่กับ &
        MsgBox("Login Name: " & TextBox1.Text & vbNewLine & "Password: " & TextBox2.Text)
    End Sub
    0 references
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        'เมื่อโหลดฟอร์มขึ้นมาให้แสดงข้อความเริ่มต้นดังนี้
        TextBox1.Text = "Your Login Name"
        TextBox2.Text = "##########"
    End Sub
End Class
```

**ภาพที่ 2.13**

3) รันโปรแกรมที่สร้างมา จะได้ผลลัพธ์ดังรูป




**ภาพที่ 2.14**

2.3.5.3 คอนโทรลเช็กบ็อกซ์ (CheckBox)

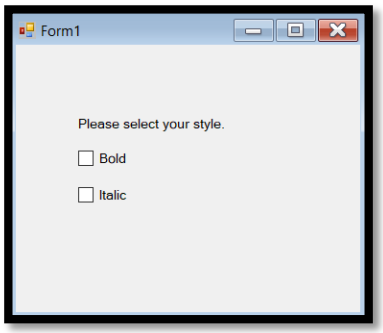
☑ คอนโทรลเช็กบ็อกซ์จะให้ผู้ใช้เลือกว่าจะเช็ก หรือไม่เช็ก เพียงอย่างใดอย่างหนึ่งเท่านั้น ค่าของคอนโทรลเช็กบ็อกซ์จะอยู่ในคุณสมบัติ Checked ที่มีค่าเป็น True หรือ False หรือคุณสมบัติ CheckState

<u>ตัวอย่างโปรแกรมแสดงการใช้งานเช็กบ็อกซ์</u>

ตัวอย่างต่อไปนี้เป็นโปรแกรมแสดงการทำงานของเช็กบ็อกซ์ โดยเราจะใช้เช็กบ็อกซ์ในการเลือกว่าต้องการตัวอักษรแบบตัวเน้น (Bold) หรือแบบตัวเอียง (Italic) จากนั้นจะปรากฏกล่องข้อความแสดงสิ่งที่เราเลือก

<u>ขั้นตอนการสร้างโปรแกรม</u>

1) สร้างโปรเจคชนิด Windows Forms App ใหม่ขึ้นมา และให้เราวางคอนโทรลต่างๆ ตามรูป




**ภาพที่ 2.15**

2) ไปที่หน้าต่าง Code Editor จากนั้นใส่คำสั่งโปรแกรมดังนี้

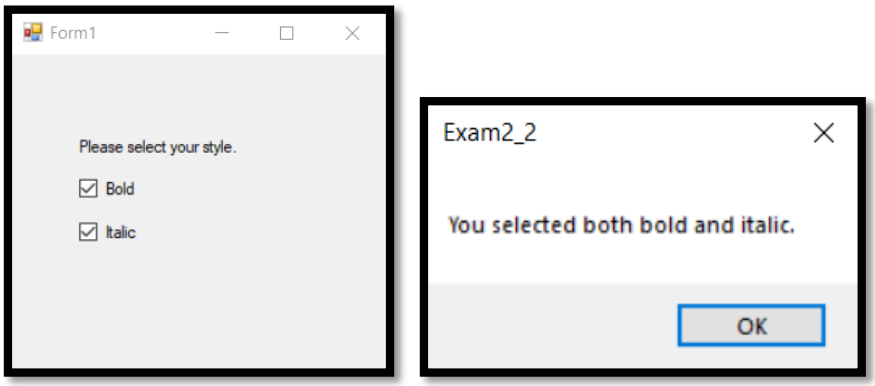
```
1 reference
Public Class Form1
    0 references
    Private Sub CheckBox1_CheckedChanged(sender As Object,
        e As EventArgs) Handles CheckBox1.CheckedChanged
        If CheckBox1.Checked = True Then 'ตรวจสอบว่า CheckBox1 ถูกเลือกหรือไม่
            If CheckBox2.Checked = True Then 'ตรวจสอบว่า CheckBox2 ถูกเลือกหรือไม่
                'ถ้าถูกเลือก
                MsgBox("You selected both bold and italic.", MsgBoxStyle.OkOnly)
            Else
                'ถ้าไม่ถูกเลือก
                MsgBox("You selected bold.", MsgBoxStyle.OkOnly)
            End If
        End If

    End Sub
```

```
    0 references
    Private Sub CheckBox2_CheckedChanged(sender As Object,
        e As EventArgs) Handles CheckBox2.CheckedChanged
        If CheckBox2.Checked = True Then 'ตรวจสอบว่า CheckBox2 ถูกเลือกหรือไม่
            If CheckBox1.Checked = True Then 'ตรวจสอบว่า CheckBox1 ถูกเลือกหรือไม่
                'ถ้าถูกเลือก
                MsgBox("You selected both bold and italic.", MsgBoxStyle.OkOnly)
            Else
                'ถ้าไม่ถูกเลือก
                MsgBox("You selected italic.", MsgBoxStyle.OkOnly)
            End If
        End If
    End Sub
End Class
```

**ภาพที่ 2.16**

3) รันโปรแกรมที่สร้างมาจะได้ผลลัพธ์ดังรูป




**ภาพที่ 2.17**

2.3.5.4 คอนโทรลเรดิโอบัตทอน (RadioButton)

 คอนโทรลเรดิโอบัตทอน ทำหน้าที่คล้ายกับเช็กบ็อกซ์ แต่คอนโทรลนี้จะสามารถเลือกได้เพียงตัวเลือกเดียวเท่านั้นในกลุ่มหนึ่ง

Example จากโจทย์เดิมข้างต้น ให้เปลี่ยนจาก CheckBox เป็น RadioButton โดยให้เลือก Bold กับ Italic เหมือนเดิม แล้วรันให้ได้ผลลัพธ์ออกมาตามที่เลือก

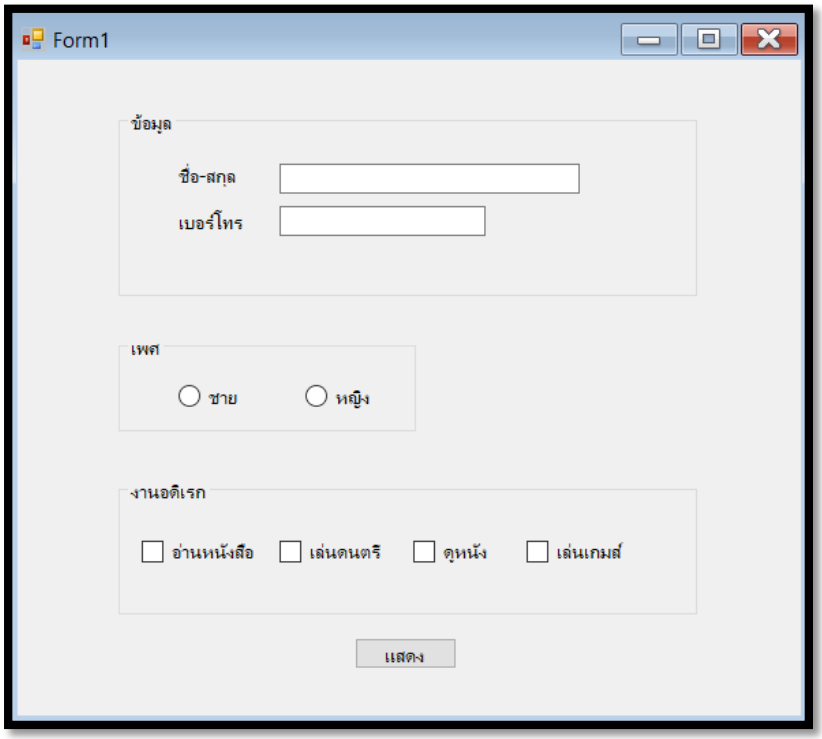
2.3.5.5 คอนโทรลกรุ๊ปบ็อกซ์ (GroupBox)

 คอนโทรลกรุ๊ปบ็อกซ์ใช้สำหรับจัดกลุ่มคอนโทรลที่ใช้ร่วมกัน ช่วยเพิ่มความเรียนร้อยและความสวยงามของหน้าตาโปรแกรมมากขึ้น เช่น ใช้แบ่งเรดิโอบัตทอน บนฟอร์มออกเป็นกลุ่มๆ เป็นต้น

ตัวอย่างโปรแกรมการใช้งานกรุ๊ปบ็อกซ์

เป็นการใช้กรุ๊ปบ็อกซ์จัดกกลุ่ม เรดิโอบัตทอน เช็กบ็อกซ์ และคอนโทรลอื่นๆ ออกแบบหน้าจอเป็นแบบฟอร์มสำหรับผู้ใช้กรอกข้อมูล และเมื่อคลิกปุ่ม Ok จะแสดงข้อมูลทั้งหมดบนกล่องข้อควา

1) สร้างโปรเจคใหม่ขึ้นมาชนิด Windows Forms App ชื่อ GroupBox และให้คอนโทรลต่างๆ ลงบนฟอร์ม พร้อมทั้งกำหนดคุณสมบัติต่างๆดังนี้



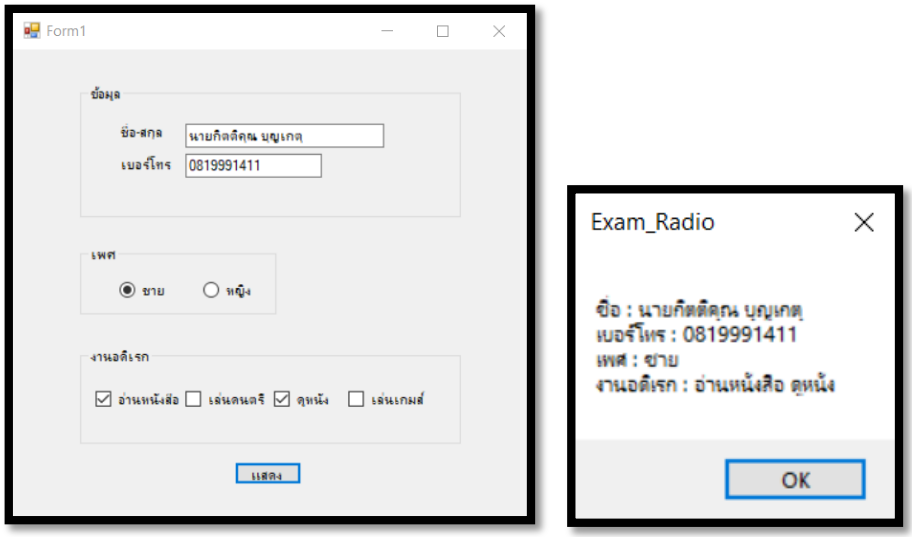

**ภาพที่ 2.18**

2) ดับเบิ้ลคลิ๊กที่ปุ่ม Button1 จะแสดงหน้าต่าง Code Editor จากนั้นใส่โค้ดโปรแกรม ดังนี้

```
Public Class Form1
    0 references
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim sex, hobby As String            'สร้างตัวแปร sex, hobby เก็บข้อความ
        If RadioButton1.Checked Then        'ถ้าเช็คเพศชายที่ RadioButton1
            sex = RadioButton1.Text         'เก็บข้อความ เพศชาย ในตัวแปร sex
        End If
        If RadioButton2.Checked Then        'ถ้าเช็คเพศหญิงที่ RadioButton2
            sex = RadioButton2.Text         'เก็บข้อความ เพศหญิง ในตัวแปร sex
        End If
        If CheckBox1.Checked Then           'ถ้าเช็กอ่านหนังสือที่ CheckBox1
            hobby = CheckBox1.Text          'เก็บข้อความอ่านหนังสือในตัวแปร hobby
        End If
        If CheckBox2.Checked Then           'ถ้าเช็กเล่นดนตรีที่ CheckBox2
            hobby += " " + CheckBox2.Text   'เก็บข้อความเพิ่มเติมในตัวแปร hobby
        End If
        If CheckBox3.Checked Then
            hobby += " " + CheckBox3.Text
        End If
        If CheckBox4.Checked Then
            hobby += " " + CheckBox4.Text
        End If
        MsgBox("ชื่อ : " & TextBox1.Text & vbNewLine _
            & "เบอร์โทร : " & TextBox2.Text & vbNewLine _
            & "เพศ : " & sex & vbNewLine _
            & "งานอดิเรก : " & hobby)
    End Sub
End Class
```

**ภาพที่ 2.19**

3) รันโปรแกรมที่สร้างมาจะได้ผลลัพธ์ ดังภาพ



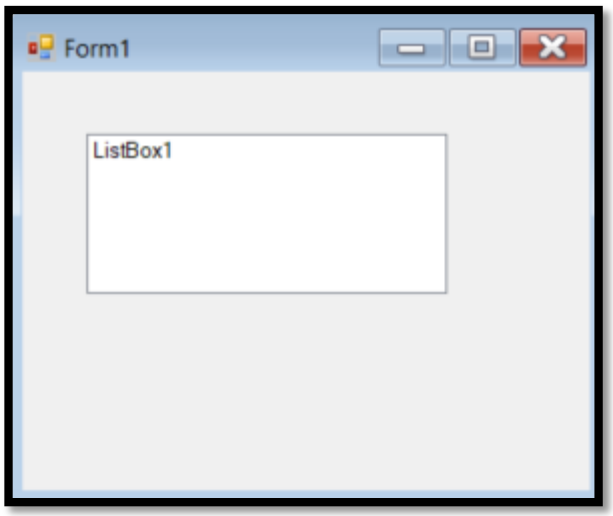

**ภาพที่ 2.20**

2.3.5.6 คอนโทรลลิสต์บ็อกซ์ (ListBox)

เราจะสังเกตได้ว่าทั้งเรดิโอบัตทอนและเช็กบ็อกซ์นั้น ถ้านำมาใช้แสดงตัวเลือกหลายๆตัว จะใช้เนื้อที่บนฟอร์มพอสมควร ในกรณีนี้เราสามารถนำเอาคอนโทรลลิสต์บ็อกซ์มาใช้แทนที่การแสดงตัวเลือกหลายๆ ตัวได้ในแบบของรายการ

<u>ตัวอย่างโปรแกรมการใช้งานลิสต์บ็อกซ์</u>

1) สร้างโปรเจ็กต์ใหม่ชนิด Windows Forms App ขึ้นมา กดคอนโทรลลิสต์บ็อกซ์มาวางบนฟอร์ม




**ภาพที่ 2.21**

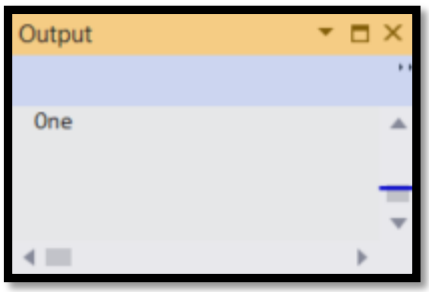
2) ไปที่หน้าต่าง Code Editor และใส่โค้ดโปรแกรมดงนี้

```
Public Class Form1
    0 references
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles Me.Load
        ListBox1.Items.Add("One")
        ListBox1.Items.Add("Two")
        ListBox1.Items.Add("Three")
        ListBox1.Items.Add("Four")
        ListBox1.Items.Add("Five")
        Console.WriteLine(ListBox1.Items(0))
    End Sub
End Class
```

**ภาพที่ 2.22**

เมื่อเปิดแสดงหน้าจอโปรแกรมจะเริ่มทำงานอีเวนต์ Form1_Load โดยเพิ่มรายการเข้าไปในลิสต์บอกซ์ด้วยคำสั่ง Item.Add() จำนวน 5 รายการ สุดท้ายจะหาค่าในรายการที่ตำแหน่ง Index(0) โดยให้พิมพ์แสดงค่าผลลัพธ์ในหน้าต่าง Output

3) คลิก Start เพื่อรันโปรแกรม จะแสดงรายการแรกสุดออกมาทางหน้าต่าง Output คือ One



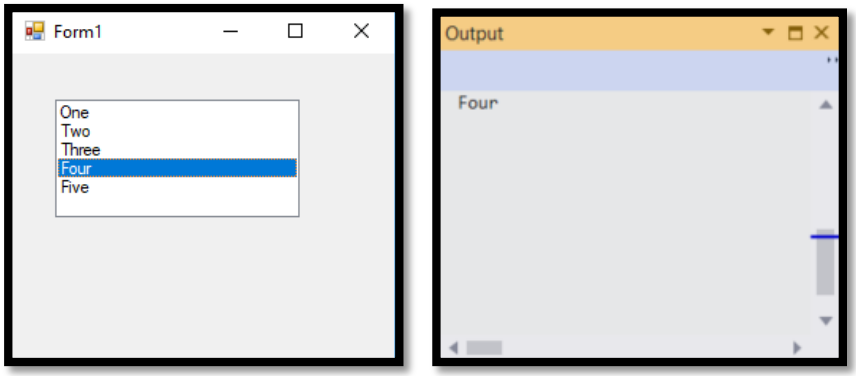

**ภาพที่ 2.23**

4) ให้เพิ่มโค้ดในส่วนอีเวนต์ ListBox1_SelectedIndexChanged เพื่อให้ทำงานเมื่อมีการเลือกรายการ โดยให้แสดงค่าจากรายการที่ถูกเลือกด้วยคุณสมบัติ Text เป็นผลลัพธ์หน้าต่าง Output ดังนี้

```
Public Class Form1
    0 references
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles Me.Load
        ListBox1.Items.Add("One")
        ListBox1.Items.Add("Two")
        ListBox1.Items.Add("Three")
        ListBox1.Items.Add("Four")
        ListBox1.Items.Add("Five")
    End Sub
    0 references
    Private Sub ListBox1_SelectedIndexChanged(sender As Object,
        e As EventArgs) Handles ListBox1.SelectedIndexChanged
        Console.WriteLine(ListBox1.Text)      'แสดงข้อความที่เลือกจาก ListBox1
    End Sub
End Class
```

**ภาพที่ 2.24**

5) เมื่อรันโปรแกรม จะแสดงรายการบนลิสต์บ็อกซ์ จากนั้นให้ผู้ใช้เลือกรายการ ก็จะนำค่าจากคุณสมบัติ Text ออกมาแสดงในหน้าต่าง Output



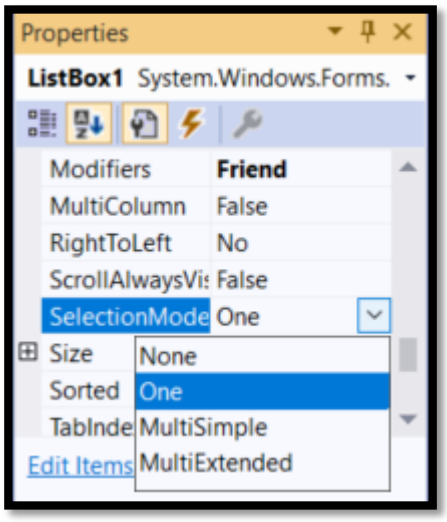

**ภาพที่ 2.25**

6) เราสามารถหาจำนวนของรายการในลิสต์บ็อกซ์ โดยใช้คุณสมบัติ Count ของลิสต์บ็อกซ์ เช่น

```
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles Me.Load
    ListBox1.Items.Add("One")
    ListBox1.Items.Add("Two")
    ListBox1.Items.Add("Three")
    ListBox1.Items.Add("Four")
    ListBox1.Items.Add("Five")
    Console.WriteLine(ListBox1.Items.Count)
End Sub
```

**ภาพที่ 2.26**

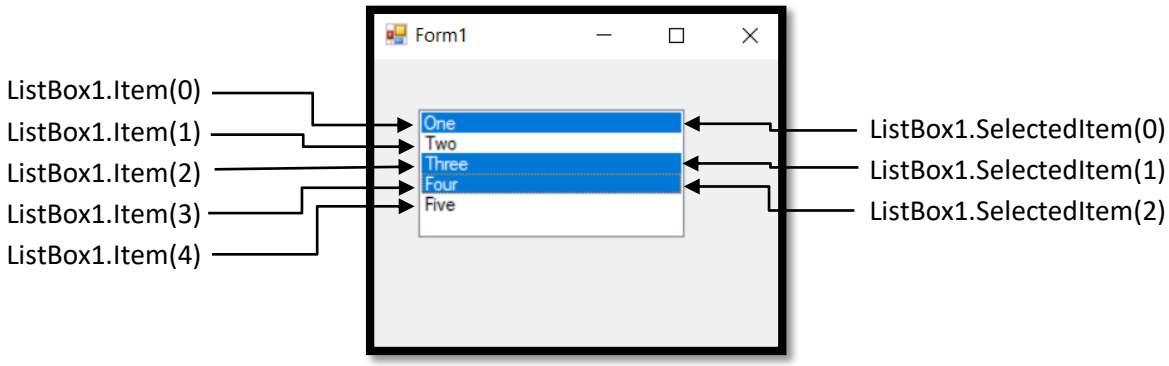
7) เราสามารถกำหนดว่าจะให้ผู้ใช้เลือกรายการในลิสต์บ็อกซ์ได้ทีละรายการหรือหลายรายการ โดยกำหนดที่คุณสมบัติ SelectionMode ซึ่งแต่ละค่าจะมีความหมายดังนี้




**ภาพที่ 2.27**

- None ไม่สามารถเลือกได้ หมายถึง ลิสต์บ็อกซ์ที่แสดงอย่างเดียว ไม่สามารถเลือกได้
- One เลือกได้รายการเดียว เป็นลิสต์บ็อกซ์แบบทั่วไป
- MultiSimple เลือกได้หลายรายการแบบ 1 กด <Spacebar> จะเป็นการเลือกรายการในลิสต์บ็อกซ์
- MultiExtended เลือกได้หลายรายการแบบ 2 โดยมีวิธีการกดปุ่มเลือกอยู่ 2 แบบ คือ
  - ✓ กดปุ่ม <Shift> พร้อมกับการคลิก จะเป็นการเลือกรายการที่อยู่ระหว่างที่เลือกไว้ก่อนหน้ากับรายการที่เลือกไว้ในปัจจุบัน
  - ✓ กดปุ่ม <Ctrl> พร้อมกับการคลิกรายการที่ต้องการ จะเป็นการเลือกรายการต่างๆ ในลิสต์บ็อกซ์

สำหรับลิสต์บ็อกซ์ที่เลือกได้หลายรายการ เราสามารถทราบว่าผู้ใช้ได้เลือกรายการใด โดยการตรวจสอบคุณสมบัติ SelectedItems ซึ่งเป็นคอลเล็กชันที่เก็บออบเจ็กต์ Item ที่ผู้ใช้เลือกไว้ นอกจากนี้ยังมีคุณสมบัติ SelectedIndices ที่เก็บอินเด็กซ์รายการแต่ละตัวที่ผู้ใช้เลือกไว้ในลิสต์บ็อกซ์ ดังรูป




**ภาพที่ 2.28**

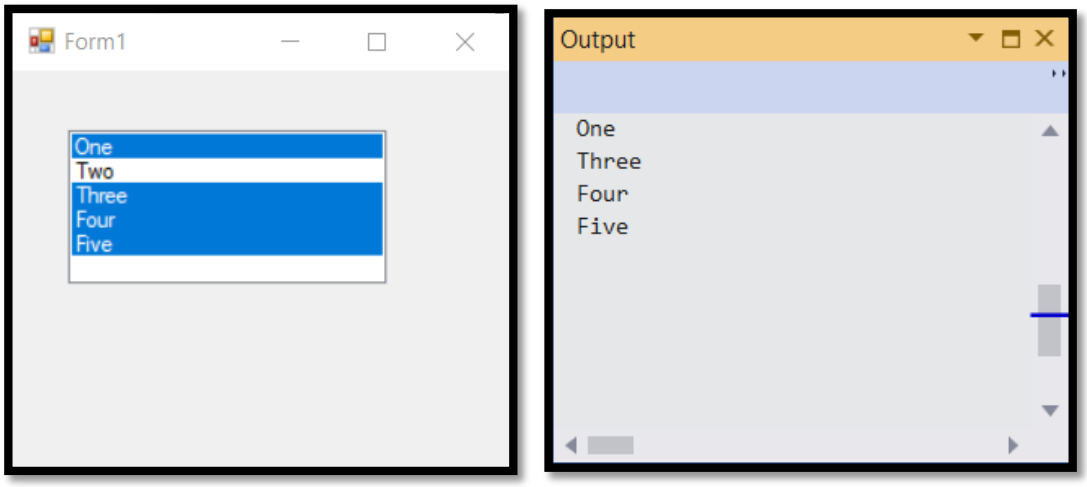
<u>สรุป</u> Items รายการตัวเลือก SelectedItems ก็คือ กลุ่มของ Items ที่ถูกเลือก ส่วน Items ที่ไม่เลือกจะไม่อยู่ในกลุ่มนี้ ดังนั้น SelectedItems จึงเป็นซับเซตของ Items เสมอ จากภาพ 2.28 จะเห็นว่า Items จะมี Index 5 ตำแหน่ง แต่ SelectedItems จะมี Index เพียง 3 ตำแหน่ง โดยค่าของ Index ใน Items และ SelectedItems จะอ้างอิงถึงรายการที่แตกต่างกัน เช่น Index ค่า 1 ใน Items คือรายการ Two แต่ค่าใน SelectedItems จะเป็นรายการ Three ซึ่งจุดนี้เป็นจุดที่ทำให้สับสนและงงกันมาก

<u>เพิ่มเติม</u> จาก Code เดิม ให้กำหนดคุณสมบัติของ SelectionMode ของลิสต์บ็อกซ์ในหน้าต่างคุณสมบัติเป็น MultiExtended จากนั้นไปที่หน้าต่าง Code Editor แล้วสร้างโค้ดอีเวนต์ SelectionIndexChanged ของ ListBox1 ให้ทำงานเมื่อผู้ใช้เลือกรายการโดยจะแสดงรายการที่ถูกเลือกด้วยคุณสมบัติ SelectedItem ดังนี้

```
Public Class Form1
    0 references
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles Me.Load
        ListBox1.Items.Add("One")
        ListBox1.Items.Add("Two")
        ListBox1.Items.Add("Three")
        ListBox1.Items.Add("Four")
        ListBox1.Items.Add("Five")
        Console.WriteLine(ListBox1.Items.Count)
    End Sub
    0 references
    Private Sub ListBox1_SelectedIndexChanged(sender As Object,
        e As EventArgs) Handles ListBox1.SelectedIndexChanged
        Dim i As Integer
        For i = 0 To ListBox1.SelectedItems.Count - 1
            Console.WriteLine(ListBox1.SelectedItems(i))
        Next i
    End Sub
End Class
```

**ภาพที่ 2.29**

เมื่อรันโปรแกรม จะแสดงรายการทั้งหมดที่ผู้ใช้เลือกบนหน้าต่าง Output




**ภาพที่ 2.30**

2.3.5.7 คอนโทรลคอมโบบ็อกซ์ (ComboBox)

คอมโบบ็อกซ์ จะรวมความสามารถของเท็กซ์บ็อกซ์และลิสต์บ็อกซ์เข้าไว้ด้วยกัน ทำให้เราสามารถเลือกรายการได้เช่นเดียวกับลิสต์บ็อกซ์ และแก้ไขรายการได้เหมือนกับเท็กซ์บ็อกซ์ แต่ใช้พื้นที่น้อยกว่าลิสต์บ็อกซ์ และใช้คำสั่งในการทำงานคล้ายกัน ดังนั้นเราสามารถนำความรู้เกี่ยวกับลิสต์บ็อกซ์มาใช้กับคอมโบบ็อกซ์ได้

การกำหนดค่ารูปแบบของคอมโบบ็อกซ์ เราจะกำหนดผ่านทางคุณสมบัติ DropDownStyle โดยคอมโบบ็อกซ์มีอยู่ 3 รูปแบบ ได้แก่

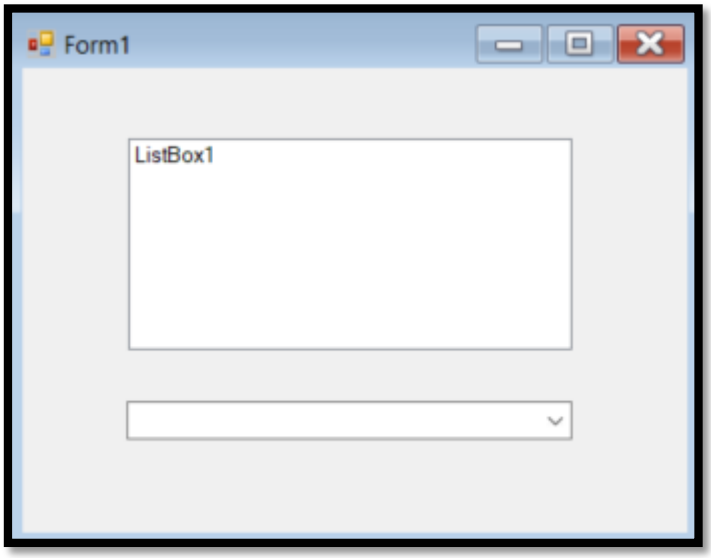
- ซิมเปิลคอมโบบ็อกซ์ (Simple ComboBox) เป็นคอมโบบ็อกซ์ที่แสดงรายการอยู่ตลอดเวลา และเมื่อจำนวนรายการเกินความสูงของคอนโทรล คอนโทรลจะแสดงสกอลบาร์แนวตั้ง เพื่อใช้เลื่อนดูรายการที่เหลือได้

- ดร็อปดาวน์คอมโบบ็อกซ์ (Dropdown ComboBox) เป็นคอมโบบ็อกซ์ที่ผู้ใช้สามารถใส่ค่าลงไปได้โดยตรง หรือเลือกจากรายการที่มีอยู่ก็ได้ โดยการคลิกที่ปุ่มลูกศรลง เมื่อเลือกตัวเลือกใดในรายการแล้ว ตัวเลือกนั้นจะแสดงอยู่ช่องบนสุดของคอนโทรล

- ดร็อปดาวน์ลิสต์บ็อกซ์ (Dropdown ListBox) เป็นคอมโบบ็อกซ์ที่คล้ายกับลิสต์บ็อกซ์ แต่จะประหยัดเนื้อที่ที่ใช้แสดงรายการมากกว่า โดยที่ผู้ใช้ไม่สามารถใส่ค่าเองได้ จะเลือกข้อมูลจากรายการที่มีอยู่ได้เท่านั้น

ตัวอย่างการใช้งานโปรแกรมลิสต์บ็อกซ์และคอมโบบ็อกซ์

1) สร้างโปรเจคใหม่ชนิด Windows Forms App ขึ้นมา และให้วางคอนโทรลต่างๆ ลงบนฟอร์ม พร้อมทั้งกำหนดคุณสมบัติต่างๆ ของฟอร์ม ดังภาพ




**ภาพที่ 2.31**

2) ดับเบิ้ลคลิกที่พื้นที่ว่างบนฟอร์ม จะได้ Events Load ขึ้นมา และเข้าสู่หน้า Code Editor จะทำการสร้างโค้ดอีเวนต์ Form1_Load เอาไว้ แล้วให้เราใส่ค่าเริ่มต้นของลิสต์บ็อกซ์และคอมโบบ็อกซ์เพิ่มเติม ดังนี้

```
    0 references
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        ListBox1.Items.Add("One")
        ListBox1.Items.Add("Two")
        ListBox1.Items.Add("Three")
        ListBox1.Items.Add("Four")
        ListBox1.Items.Add("Five")
        ComboBox1.Items.Add("One")
        ComboBox1.Items.Add("Two")
        ComboBox1.Items.Add("Three")
        ComboBox1.Items.Add("Four")
        ComboBox1.Items.Add("Five")
    End Sub
End Class
```

**ภาพที่ 2.32**

3) คลิกที่ลิสต์บ็อกซ์ จากนั้นไปที่หน้าต่างคุณสมบัติและแท็บ Events ให้ดับเบิ้ลคลิ๊กที่อีเวนต์ SelectedIndexChanged จะเข้าสู่หน้าต่าง Code Editor สำหรับเขียนโปรแกรม จากนั้นให้ใส่โค้ดแสดงข้อความจากรายการที่ผู้ใช้เลือกในลิสต์บ็อกซ์ปรากฏในกล่องข้อความ

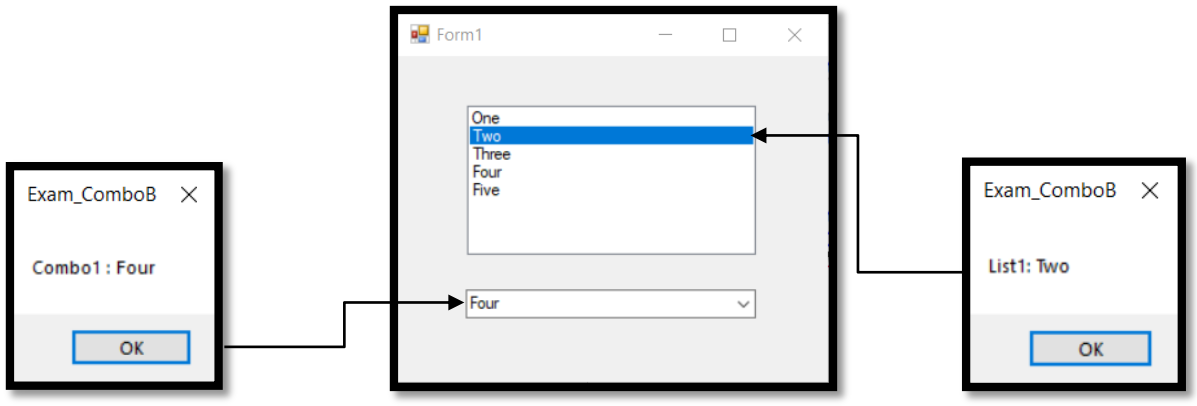
4) คลิกที่คอมโบบ็อกซ์ จากนั้นไปที่หน้าต่างคุณสมบัติและแท็บ Events ให้ดับเบิ้ลคลิกที่อีเวนต์ TextChanged จะเข้าสู่หน้าต่าง Code Editor สำหรับเขียนโปรแกรม จากนั้นให้ใส่โค้ดแสดงข้อความจากรายการที่ผู้ใช้เลือกในคอมโบบ็อกซ์ให้ปรากฏในกล่องข้อความ

```
    0 references
    Private Sub ListBox1_SelectedIndexChanged(sender As Object,
        e As EventArgs) Handles ListBox1.SelectedIndexChanged
        MsgBox("List1: " & ListBox1.Text, MsgBoxStyle.OkOnly)
    End Sub

    0 references
    Private Sub ComboBox1_TextChanged(sender As Object,
        e As EventArgs) Handles ComboBox1.TextChanged
        MsgBox("Combo1 : " & ComboBox1.Text, MsgBoxStyle.OkOnly)
    End Sub
End Class
```

**ภาพที่ 2.33**

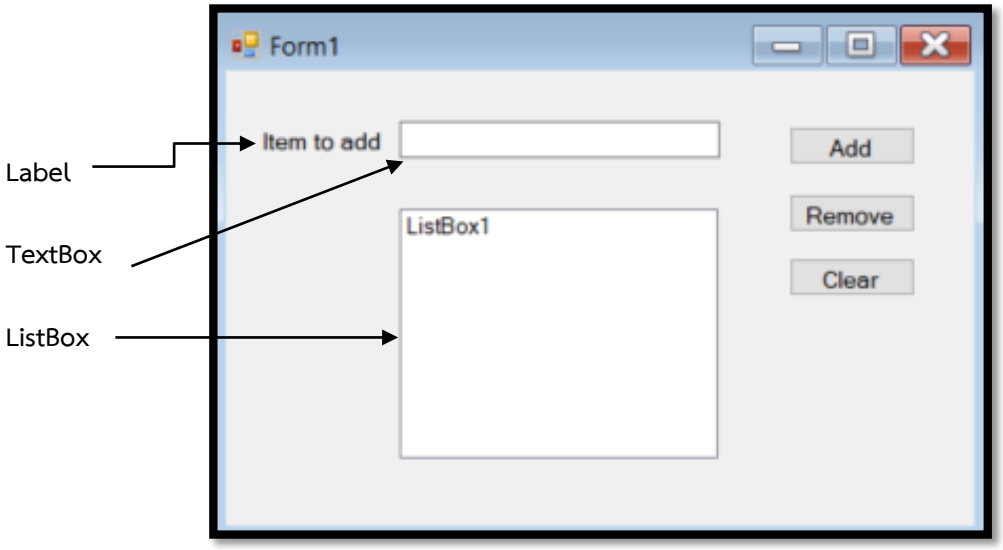
5) รันโปรแกรม จะแสดงผลลัพธ์ดังภาพ




**ภาพที่ 2.34**

ตัวอย่างโปรแกรมแสดงการเพิ่มและลบรายการในลิสต์บ็อกซ์/คอมโบบ็อกซ์

1) สร้างโปรเจคใหม่ชนิด Windows Forms App ขึ้นมา และให้วางคอนโทรลต่างๆ บนฟอร์ม พร้อมกำหนดคุณสมบัติ ดังภาพ




**ภาพที่ 2.35**

2) เขียนโปรแกรมให้กับปุ่ม Add โดยดับเบิลคลิกปุ่ม Add จะปรากฏหน้าต่าง Code Editor และเกิดอีเวนต์ Button1_Click จากนั้นให้ใส่โค้ดรับข้อมูลจากเท็กซ์บ็อกซ์มาเพิ่มเป็นรายการอยู่ในลิสต์บ็อกซ์

3) เขียนโปรแกรมให้กับปุ่ม Remove โดยดับเบิลคลิกปุ่ม Remove จะเกิดอีเวนต์ Button2_Click จากนั้นให้ใส่โค้ดตรวจสอบว่าผู้ใช้ได้เลือกรายการในลิสต์บ็อกซ์เพื่อจะลบออกหรือไม่ ถ้ามีจะถอดรายการที่เลือกออก ถ้าไม่มีจะแจ้งผ่านกล่องข้อความว่าไม่สามารถถอดรายการออกได้

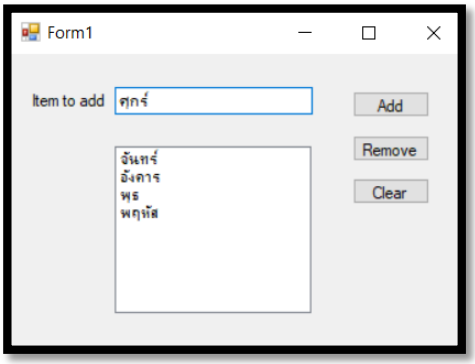
4) เขียนโปรแกรมให้กับปุ่ม Clear โดยดับเบิลคลิกปุ่ม Clear จะเกิดอีเวนต์ Button3_Click จากนั้นให้ใส่โค้ดลบรายการทั้งหมดออกจากลิสต์บ็อกซ์ ดังภาพ

```
Public Class Form1
    0 references
    Private Sub Button1_Click(sender As Object,
        e As EventArgs) Handles Button1.Click
        If TextBox1.Text <> "" Then         'เช็คว่ามีข้อความถูกพิมพ์หรือเปล่า
            ListBox1.Items.Add(TextBox1.Text) 'ถ้ามี เพิ่มข้อความจากเท็กซ์บ็อกซ์ไปที่ลิสต์บ็อกซ์
        Else
            MsgBox("Please input message!!!")  'ถ้าไม่มี ให้ขึ้นโชว์ข้อความ
        End If
        TextBox1.Text = ""          'ทำการเคลียร์ข้อความหลัง Add
        TextBox1.Focus()            'โฟกัสมายังTextBox
    End Sub
    0 references
    Private Sub Button2_Click(sender As Object,
        e As EventArgs) Handles Button2.Click
        If ListBox1.SelectedIndex > -1 Then
            'ตรวจสอบว่ามีรายการถูกเลือกอยู่หรือไม่
            'ถ้ามีรายการถูกเลือกอยู่ ลบรายการนั้นทิ้ง
            ListBox1.Items.RemoveAt(ListBox1.SelectedIndex)
        Else
            'ถ้าไม่มีรายการถูกเลือก แจ้งว่าลบไม่ได้
            MsgBox("Can't remove item from list.")
        End If
    End Sub

    0 references
    Private Sub Button3_Click(sender As Object,
        e As EventArgs) Handles Button3.Click
        ListBox1.Items.Clear()  'ลบรายการในลิสต์บ็อกซ์ทั้งหมด
    End Sub
End Class
```

**ภาพที่ 2.36**

5) ทำการคลิกรันโปรแกรม ทดสอบผลลัพธ์ที่ได้ดังภาพ



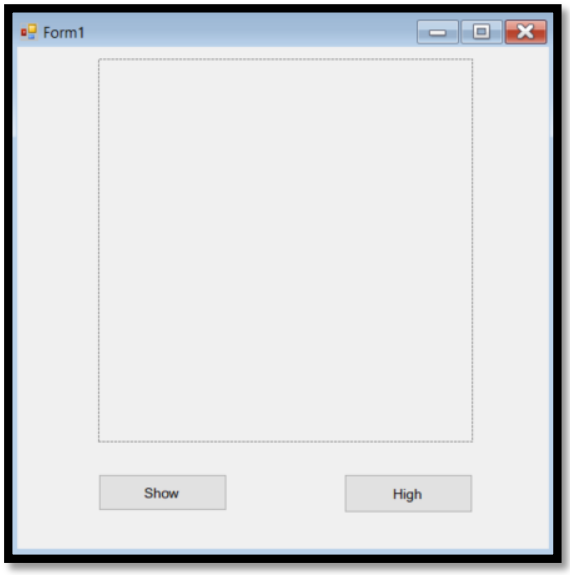

**ภาพที่ 2.37**

2.3.5.8 คอนโทรลพิกเจอร์บ็อกซ์ (PictureBox)

 เป็นคอนโทรลที่ใช้แสดงภาพและยังสร้างเป็นภาพแบ็กกราวด์ให้กับฟอร์มได้ด้วย ซึ่งการนำรูปภาพ ไอคอน หรือภาพสัญลักษณ์ต่างๆ มาใช้บนฟอร์มจะช่วยสื่อความหมายผู้ใช้งานได้ดียิ่งขึ้น

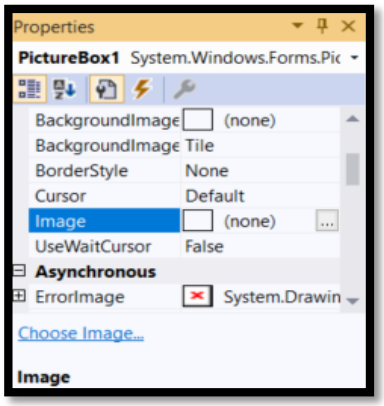
<u>ตัวอย่างโปรแกรมการเพิ่มและซ่อนภาพในพิกเจอร์บ็อกซ์</u>

1) สร้างโปรเจคชนิดใหม่เป็นชนิด Windows Forms App ขึ้นมา จากนั้นให้วางคอนโทรลพิกเจอร์บ็อกซ์และปุ่มบนฟอร์ม ตามภาพ




**ภาพที่ 2.38**

2) คลิกเลือกคอนโทรลพิกเจอร์ ไปที่ Properties และคลิกคุณสมบัติ Image เพื่อเข้าไปเลือกรูปภาพที่จะนำมาแสดงในคอนโทรลพิกเจอร์ โดยให้นำภาพมาเก็บไว้ใน Resources เพราะเมื่อเราบิลด์โปรแกรมจะแพ็กเกจไฟล์ภาพนี้ไปด้วย ทำให้สามารถรันโปรแกรมแดสงภาพบนเครื่องต่างๆ ได้อย่างถูกต้อง




**ภาพที่ 2.39**

3) เขียนโปรแกรมให้กับปุ่ม Show และ Hide

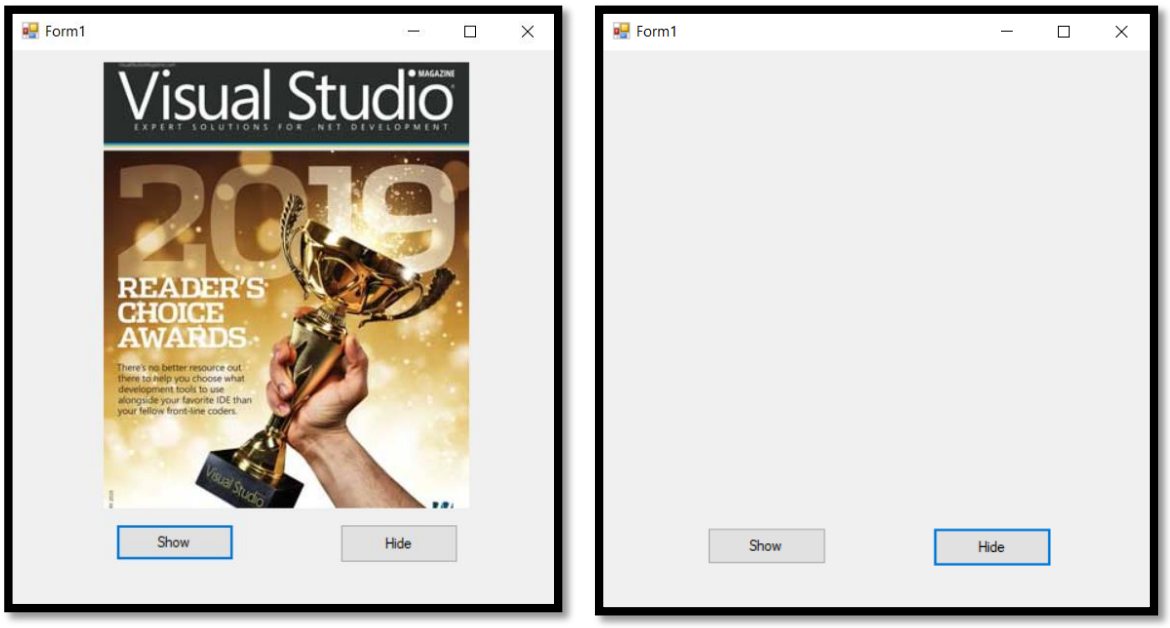
```
Public Class Form1

    0 references
    Private Sub Button1_Click(sender As Object, e As EventArgs) _
        Handles Button1.Click
        PictureBox1.Visible = True
    End Sub

    0 references
    Private Sub Button2_Click(sender As Object, e As EventArgs) _
        Handles Button2.Click
        PictureBox1.Visible = False
    End Sub
End Class
```

**ภาพที่ 2.40**

4) รันโปรแกรม จะได้ผลลัพธ์ตามภาพ




**ภาพที่ 2.41**

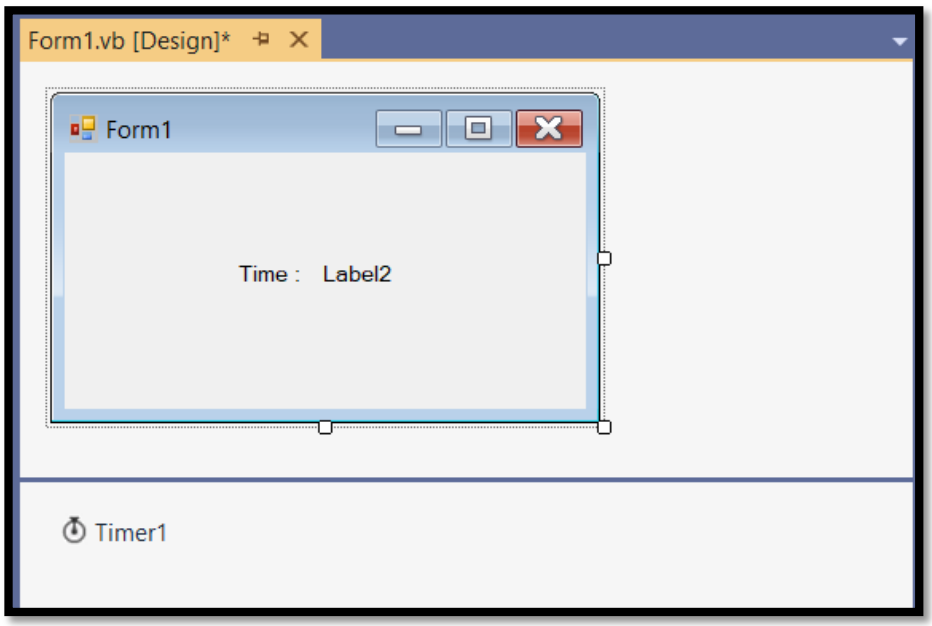
2.3.5.8 คอมโพเนนต์ไทเมอร์ (Timer)

การใช้คอมโพเนนต์ไทเมอร์จะทำให้มีการทำงานบางอย่างในทุกๆ ช่วงเวลาที่เรากำหนด ซึ่งจะช่วยให้เราสามารถทำอย่างอื่น โดยทิ้งโปรแกรมนี้ให้ทำงานตามเวลาที่ตั้งไว้ คอมโพเนนต์ไทเมอร์จัดเป็นคอนโทรลประเภทที่เรียกว่า Non Visual interface หมายถึง จะไม่เห็นตัวคอนโทรลในขณะที่รันโปรแกรมขึ้นมา ซึ่งจะพบได้บ่อยในการประยุกต์สร้างเกม แบบทดสอบ หรือการจำกัดเวลาในการแสดงหน้าจอการทำงานบางอย่าง

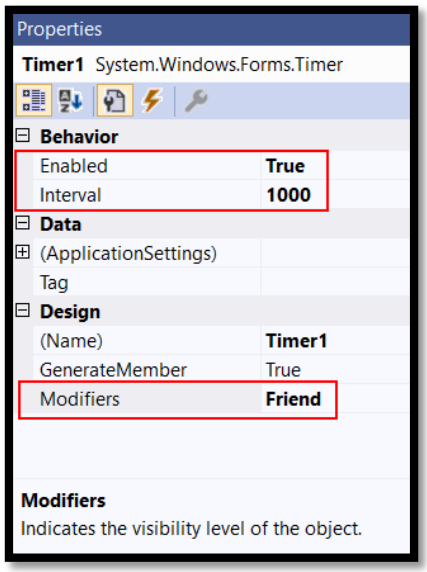
- อีเวนต์ที่สำคัญของ Timer คือ Tick เป็นอีเวนต์ที่เกิดขึ้นในทุกช่วงเวลาที่เราได้กำหนดไว้ในคุณสมบัติ Interval
- เมธอดของไทเมอร์ Start() คือเริ่มจับเวลา และ Stop() คือหยุดจับเวลา

<u>ตัวอย่างโปรแกรมการแสดงเวลาโดยใช้ไทเมอร์</u>

1) สร้างโปรเจ็กต์ใหม่ขึ้นมา จากนั้นวางคอนโทรลและคอมโพเนนต์บนฟอร์ม และกำหนดคุณสบัติของไทเมอร์ ดังภาพ




**ภาพที่ 2.42**




**ภาพที่ 2.43**

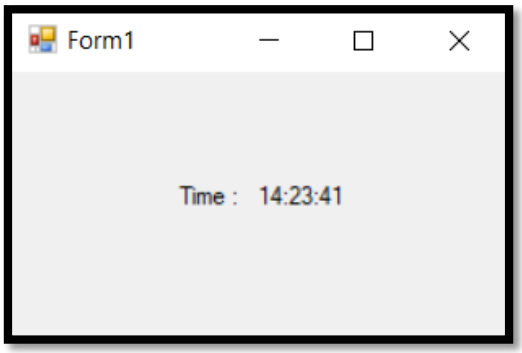
2) ดับเบิ้ลคลิกที่คอมโพเนนต์ Timer เข้าสู่หน้าต่าง Code Editor ใส่คำสั่งดังนี้

```
    0 references
    Private Sub Timer1_Tick(sender As Object, e As EventArgs) _
        Handles Timer1.Tick    'จะทำงานทุกๆ 1 วินาทีตามค่าที่กำหนดใน Interval = 1000
        Label2.Text = My.Computer.Clock.LocalTime.ToLongTimeString
    End Sub
End Class
```

**ภาพที่ 2.44**

My.Computer.Clock.LocalTime.ToLongTimeString คือ การดึงเวลาปัจจุบันบนคอมพิวเตอร์เครื่องนั้นออกมาอยู่ในรูปของข้อความ

3) คลิกรันโปรแกรมจะได้ผลลัพธ์ตามภาพ

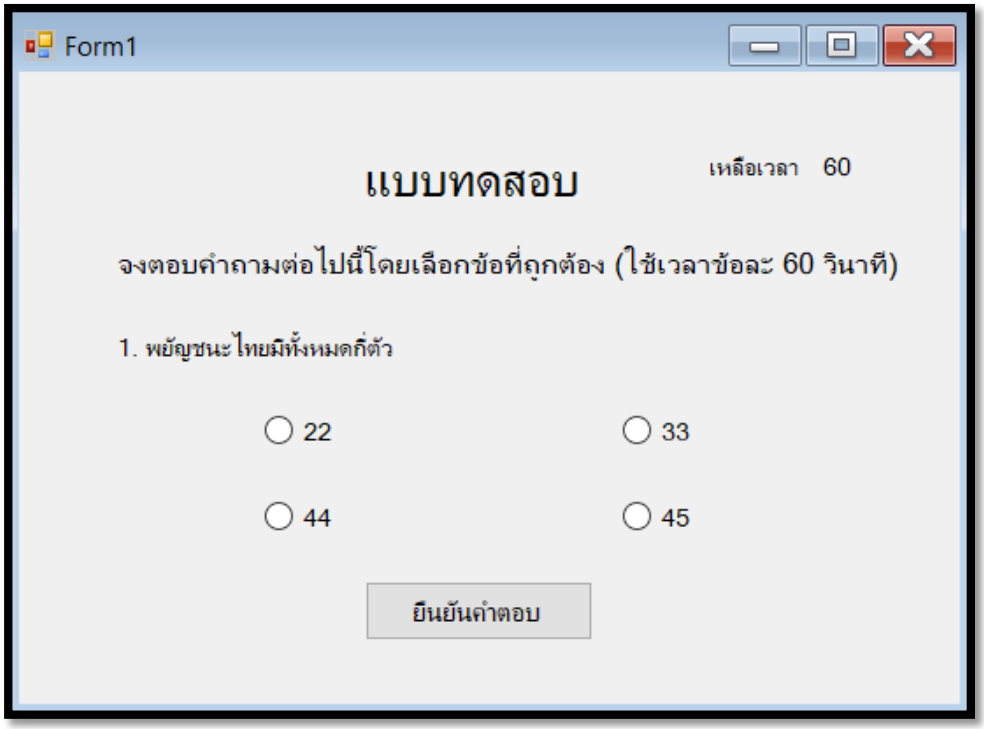



**ภาพที่ 2.45**

ตัวอย่างการทำแบบทดสอบแบบจับเวลา

โปรแกรมต่อไปนี้จะแสดงการใช้งานคอมโพเนนต์ไทเมอร์ โดยจะใช้เมธอด Start() เริ่มจับเวลา เมื่อมีการตอบคำถาม และเมธอด Stop() จะหยุดการจับเวลา และหากไม่สามารถตอบในเวลาที่กำหนด จะเรียกใช้เมธอด Stop() เพื่อหยุดการจับเวลา

1) สร้างโปรเจ็กต์ใหม่ขึ้นมา จากนั้นวางคอนโทรลและคอมโพเนนต์ต่างๆ ดังภาพ




**ภาพที่ 2.46**

2) ไปที่หน้าต่าง Code Editor กำหนดตัวแปร count = 60 เป็นเวลาในการทำแบบทดสอบหนึ่งข้อ และตัวแปล time เก็บจำนวนเวลาที่ใช้ในการทำแบบทดสอบ

3) คลิกพื้นที่ว่างบนฟอร์ม จากนั้นไปที่หน้าต่างคุณสมบัติและแท็บ Events ให้ดับเบิลคลิกี่อีเวนต์ Load จะเข้าสู่หน้าต่าง Code Editor สำหรับเขียนโค้ดการทำงานในหน้า Load ของ Form1 โดยให้กำหนด Interval ค่าการจับเวลาและเรียกใช้เมธอด Start() ให้ไทเมอร์เริ่มจับเวลา

```
Public Class Form1
    Private count As Integer = 60
    Private time As Integer
    0 references
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Timer1.Interval = 1000
        Timer1.Start()
    End Sub
End Class
```

**ภาพที่ 2.47**

4) ดับเบิลคลิกที่ปุ่มกด จะแสดงหน้าต่าง Code Editor จากนั้นแทรกโค้ดหยุดนับเวลา ตรวจคำตอบ แสดงผลคำตอบและเวลาที่ใช้ไป

5) ดับเบิลคลิกที่คอมโพเนนต์ไทเมอร์ในกรอบด้านล่าง จะแสดงหน้าต่าง Code Editor และเกิดอีเวนต์ Timer1_Tick จากนั้นให้แทรกโค้ดสำหรับแสดงการนับเวลาถอยหลัง หยุดจับเวลาเมื่อหมดเวลา และแสดงกล่องข้อความแจ้งว่าผู้ใช้ไม่สามารถตอบคำถามในเวลาที่กำหนด

```
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Timer1.Stop()
        If (RadioButton2.Checked) Then  'เมื่อตอบถูก
            time = 60 - count
            MsgBox("คำตอบถูกต้อง คุณใช้เวลาตอบ" & time & "วินาที")
        Else
            MsgBox("คุณตอบผิด")      'เมื่อตอบผิด
        End If
    End Sub

    0 references
    Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
        Label5.Text = count     'แสดงตัวเลขการนับเวลาที่คอนโทรล Label5 เป็นตัวอักษร
        count -= 1              'ลดการนับเวลาลง 1
        If (count < 0) Then
            Timer1.Stop()        'หยุดการนับเวลา
            MsgBox("คุณไม่สามารถตอบในเวลาที่กำหนด")     'แจ้งเมื่อตอบไม่ทันในเวลาที่กำหนด
        End If
    End Sub
End Class
```

**ภาพที่ 2.48**

6) รันเพื่อทดสอบโปรแกรม

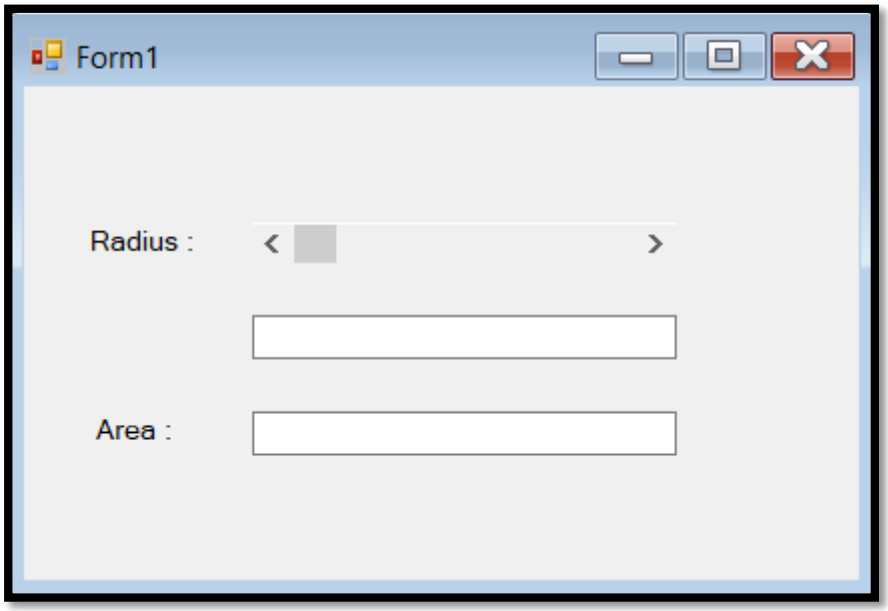
2.3.5.9 คอนโทรลสกรอลบาร์ (ScrollBar)

สกรอลบาร์จะใช้ในโอกาสที่ไม่สามารถแสดงผลข้อมูลทั้งหมดได้ในคราวเดียว เนื่องจากมีพื้นที่จำกัด หรืออาจจะใช้เป็นอินพุตในการปรับค่าข้อมูลก็ได้ ซึ่งคอนโทรลสกรอลบาร์จะมีทั้งในแนวนอนและแนวตั้ง ดังนี้

<u>ตัวอย่างโปรแกรมแสดงการใช้งานสกรอลบาร์</u>

ตัวอย่างโปรแกรมนี้จะแสดงการใช้งานคอนโทรลสกรอลบาร์ โดยจะรับค่ารัศมีวงกลมผ่านทางสกรอลบาร์แล้วแสดงค่ารัศมี และคำนวณพื้นที่วงกลมมาทางเท็กซ์บ็อกซ์

1) สร้างโปรเจ็กต์ใหม่ขึ้นมา จากนั้นวางคอนโทรลต่างๆ ดังต่อไปนี้




**ภาพที่ 2.49**

2) ดับเบิ้ลคลิกที่สกรอลบาร์ จะเข้าสู่หน้าต่าง Code Editor สำหรับเขียนโปรแกรม และแทรกโค้ดดังต่อไปนี้

```
Public Class Form1
    0 references
    Private Sub HScrollBar1_Scroll(sender As Object, e As ScrollEventArgs) _
        Handles HScrollBar1.Scroll
        TextBox1.Text = Format(HScrollBar1.Value, "###.00") 'แสดงรัศมีจากการเลื่อนสกรอบาร์
    End Sub
End Class
```

**ภาพที่ 2.50**

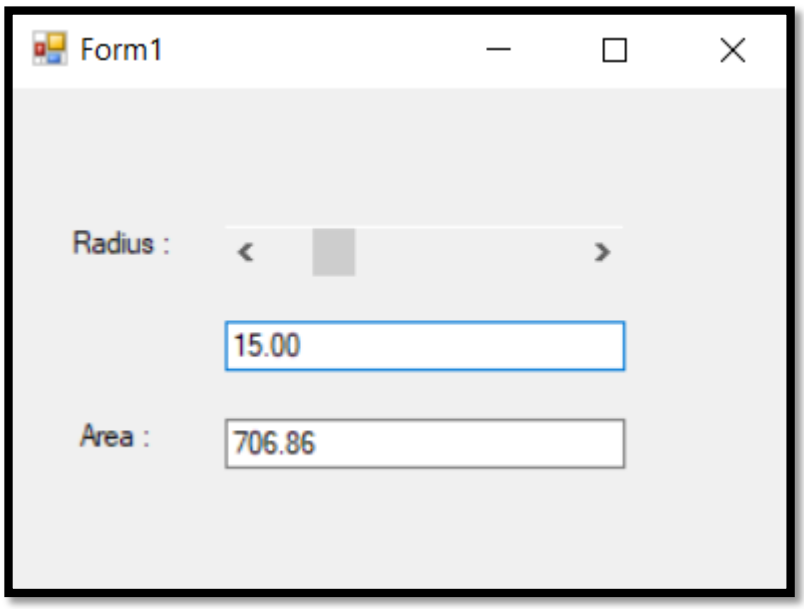
3) คลิกที่คอนโทรล HScrollBar1 จากนั้นไปที่ Properties และดูที่แท็บ Events ให้ดับเบิ้ลคลิกที่อีเวนต์ ValueChanged จะเข้าสู่หน้าต่าง Code Editor สำหรับเขียนโปรแกรม โดยให้แทรกโค้ดนำค่ารัศมีจากการปรับสกอรลบาร์มาคำนวณหาพื้นที่ของวงกลมและแสดงผลใน TextBox2

```
Public Class Form1
    0 references
    Private Sub HScrollBar1_Scroll(sender As Object, e As ScrollEventArgs) _
        Handles HScrollBar1.Scroll
        TextBox1.Text = Format(HScrollBar1.Value, "###.00") 'แสดงรัศมีจากการเลื่อนสกรอบาร์
    End Sub

    0 references
    Private Sub HScrollBar1_ValueChanged(sender As Object, e As EventArgs) _
        Handles HScrollBar1.ValueChanged 'เมื่อค่าถูกเปลี่ยน
        TextBox2.Text = Format((HScrollBar1.Value ^ 2) * Math.PI, "###.00") 'แสดงพื้นที่วงกลม
    End Sub
End Class
```

**ภาพที่ 2.51**

4) รันโปรแกรมจะได้ผลลัพธ์ดังภาพ

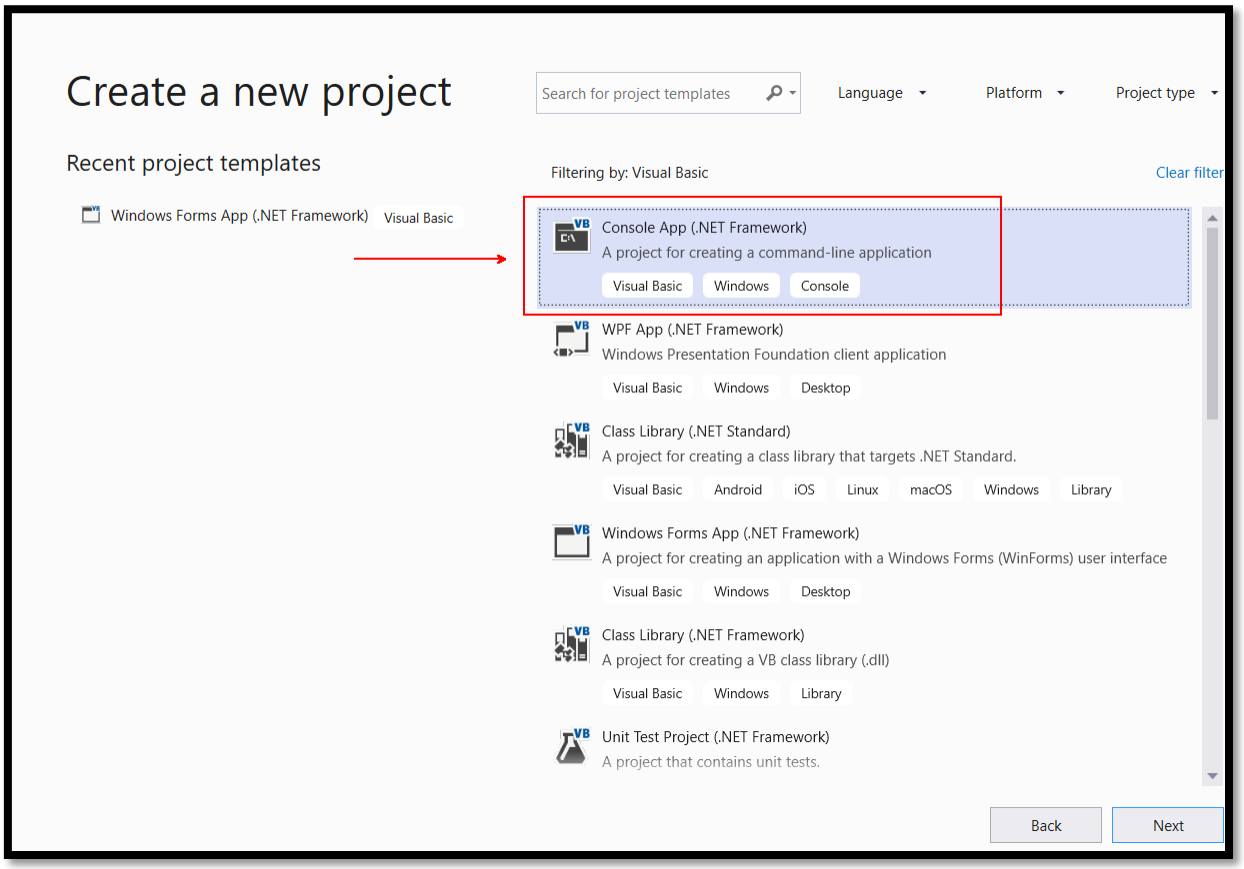



**ภาพที่ 2.52**

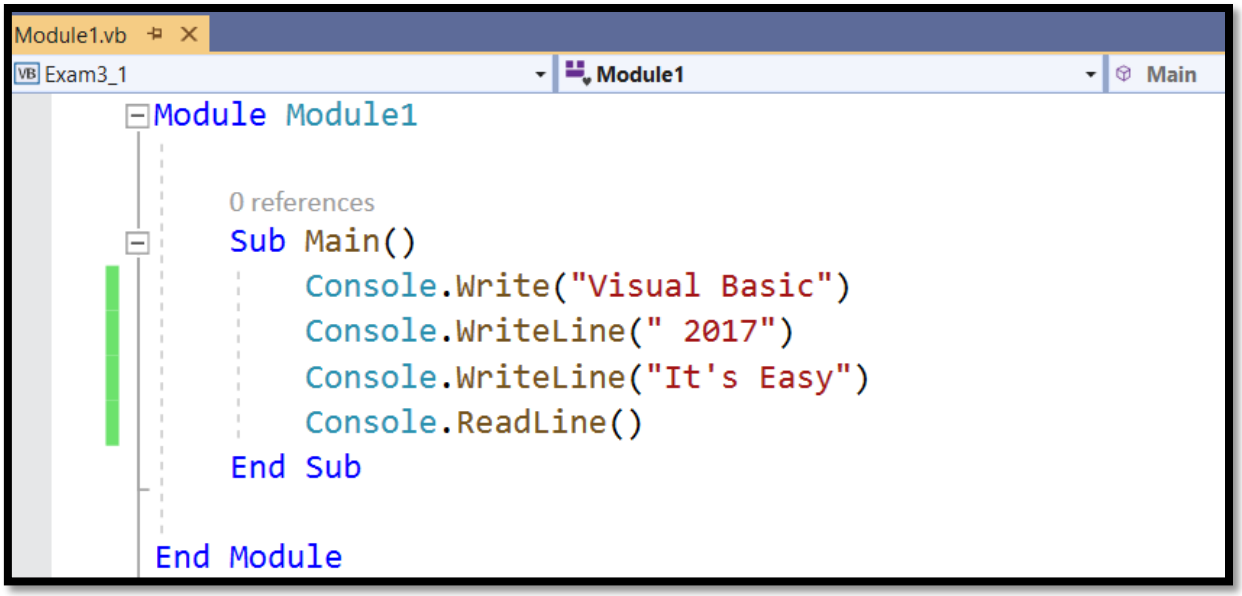
# Chapter 3
# พื้นฐานการเขียนโปรแกรม

## 3.1 การเขียนโปรแกรมแบบ Console App

การเริ่มต้นเขียนโปรแกรมอาจจะมีโค้ด 4-5 บรรทัดเท่านั้น ดังนั้นเราอาจเขียนโปรแกรมขนาดเล็กแบบเท็กซ์โหมด หรือ DOS Command line จะทำงานโดยการพิมพ์คำสั่งทีละบรรทัด เพื่อให้ง่ายต่อการศึกษาคำสั่งพื้นฐาน หรือเรียกโปรแกรมชนิดนี้ว่า Console App โดยทดลองเขียนโปรแกรมเบื้องต้นดังนี้

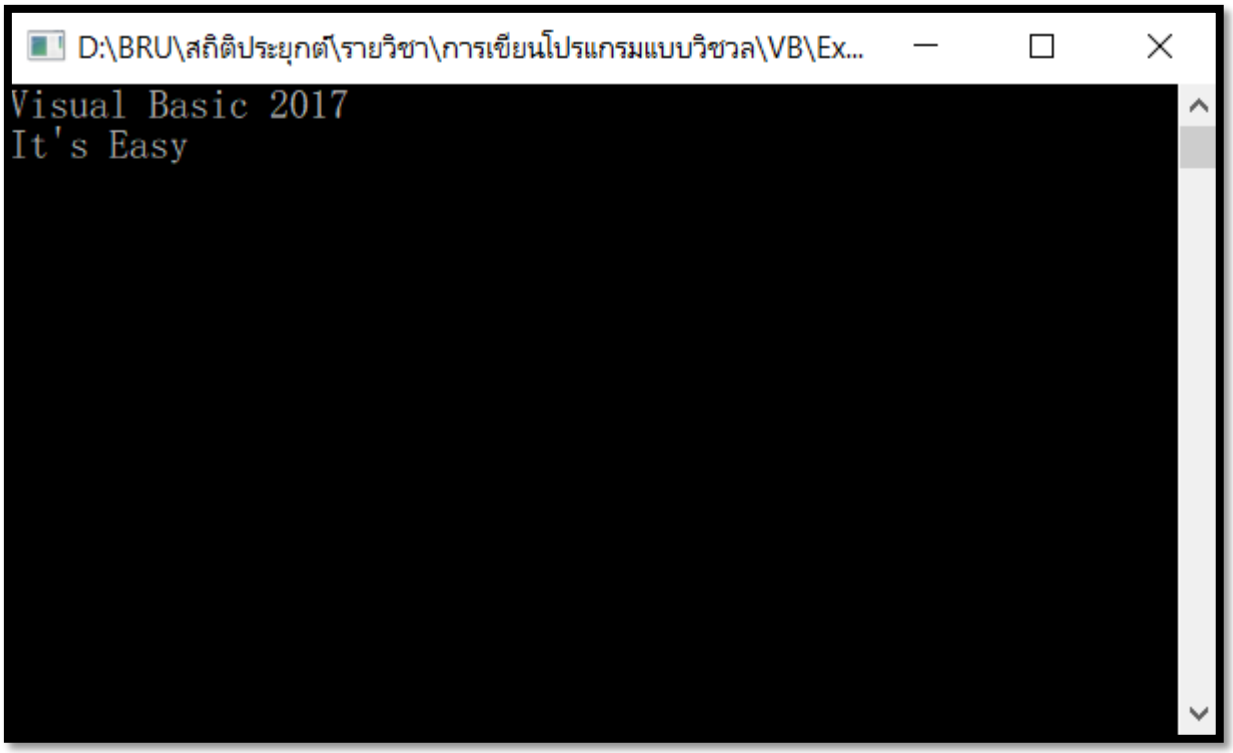



**ภาพที่ 3.1**




**ภาพที่ 3.2**

เมื่อทดลองเขียนโปรแกรมแล้วสั่งรัน จะได้ผลลัพธ์ดังภาพที่ 3.3




**ภาพที่ 3.3**

- เมธอด Write จะแสดงข้อความภายในเครื่องหมาย “ ” และแสดงเคอร์เซอร์อยู่ในบรรทัดเดิม

- เมธอด Writeline จะแสดงข้อความภายในเครื่องหมาย “ ” ตรงตำแหน่งที่เคอร์เซอร์กะพริบอยู่ จากนั้นจะเลื่อนเคอร์เซอร์ขึ้นบรรทัดใหม่
- เมธอด Readline สั่งให้โปรแกรมหยุดรอผู้ใช้กดปุ่ม <Enter> ซึ่งเราจะใส่เมธอดนี้ไว้เพื่อเปิดหน้าจอ Console แสดงผลลัพธ์ค้างไว้ หากไม่ใส่ Readline เมื่อรันโปรแกรมเสร็จหน้าจอ Console จะปิดทันที ทำให้ไม่สามารถดูผลลัพธ์ของโปรแกรมได้

## 3.2 การเขียนคำสั่งในรูปแบบต่างๆ

3.2.1 การแบ่งคำสั่งบรรทัดหนึ่งออกเป็นหลายบรรทัด

เราสามารถแยกบรรทัด ด้วยอักขระช่องว่างแล้วตามด้วยอักขระ _

```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
```

```
Private Sub Button1_Click(sender As Object, e As EventArgs) _
    Handles Button1.Click
```

**ภาพที่ 3.4**

3.2.2 การรวมคำสั่งหลายบรรทัดในบรรทัดเดียว

เราสามารถรวมหลายคำสั่งให้อยู่ในบรรทัดเดียวกัน โดยใช้เครื่องหมาย : เพื่อแยกแต่ละคำสั่งออกจากกัน ตัวอย่างเช่น

```
X = 1
Y = 2
Z = X + Y
```

```
X = 1 : Y = 2 : Z = X + Y
```

**ภาพที่ 3.5**

## 3.3 ตัวแปร ค่าคงที่ และชนิดของข้อมูล

3.3.1 ตัวแปร (Variable)

ตัวแปรใน VB มีหน้าที่เก็บข้อมูลในการทำงานของโปรแกรมไว้เป็นการชั่วคราว ตัวแปรที่เรากำหนดขึ้นมาจะต้องประกอบด้วยชื่อตัวแปรและชนิดของข้อมูล (Data Type) ที่ตัวแปรเก็บได้ เราสามารถประกาศตัวแปรได้โดยมีรูปแบบดังนี้

```
Dim <ชื่อตัวแปร> [As Type]
```

```
Dim Name As String
Dim x As Integer
```

**ภาพที่ 3.6**

<u>การตั้งชื่อตัวแปร มีกฎดังนี้</u>

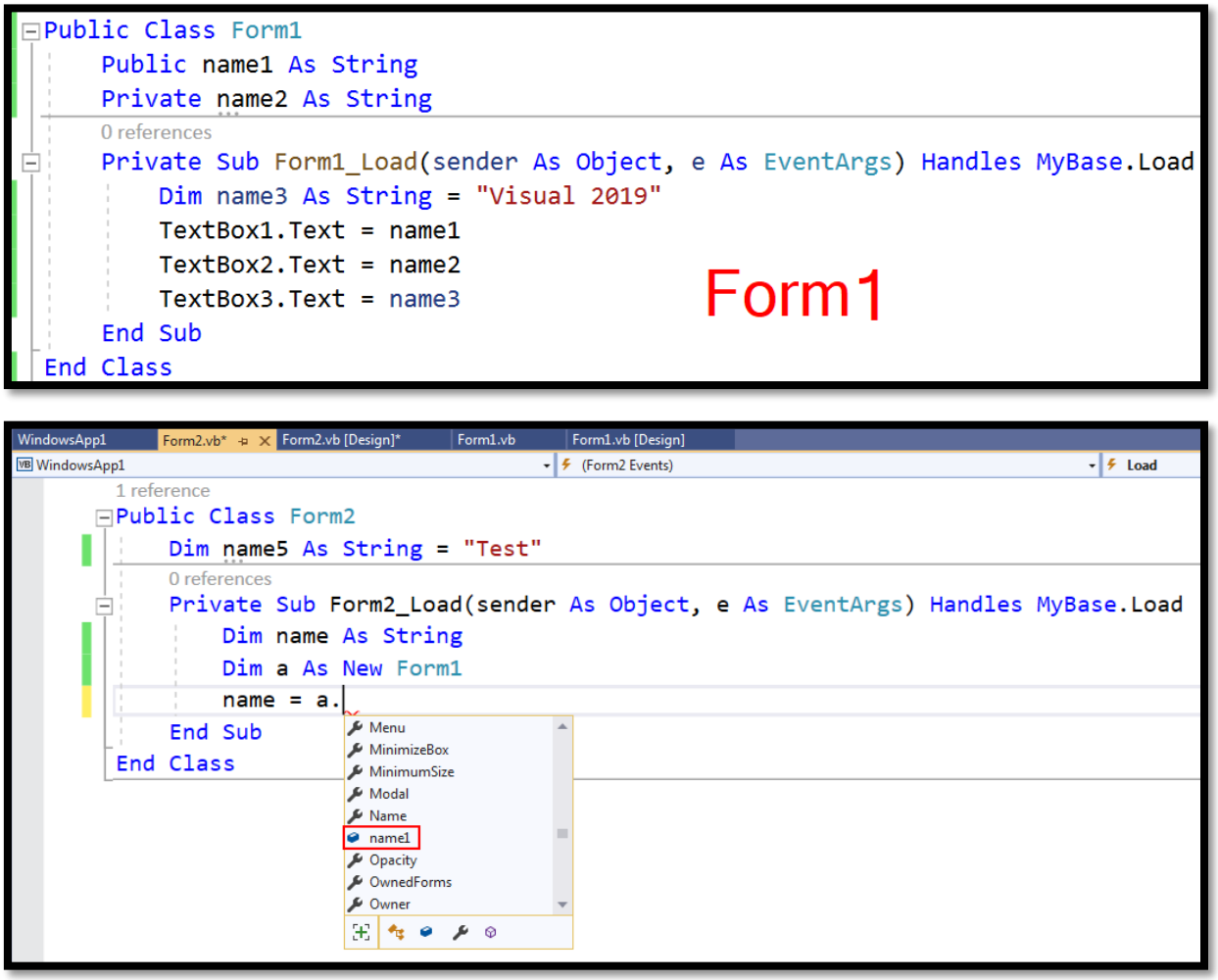
- จะต้องเริ่มต้นด้วยตัวอักขระ A-Z, a-z หรือ _
- จะต้องประกอบด้วยเฉพาะตัวอักขระ A-Z, a-z, 0-9 หรือ _
- ถ้าเริ่มต้นตัวแรกด้วย _ จะต้องตามด้วยตัวอักขระหรือตัวเลขอย่างน้อยหนึ่งตัว
- มีความยาวไม่เกิน 1023 ตัวอักษร
- ชื่อตัวแปรต้องไม่ซ้ำกันในขอบเขตเดียวกัน
- ชื่อต้องไม่ซ้ำกับคำสงวน (Reserved Word) ของ VB เช่น คำว่า Dim, If, Set ไม่ได้
- ไม่ว่าใช้ตัวอักษรพิมพ์เล็กหรือพิมพ์ใหญ่ให้ความหายเหมือนกัน

| ตัวอย่างชื่อตัวแปรที่ถูกต้อง | ตัวอย่างชื่อตัวแปรที่ผิด |
|---|---|
| Salary_12m | Salary$12 |
| _30 | 12Salary |
| _wks | _ |
| | Event |

3.3.2 ขอบเขตการประกาศตัวแปร (Scope of variable)

ในการประกาศตัวแปร เราสามารถกำหนดว่าจะให้ตัวแปรนั้นมองเห็นในส่วนใดของโปรแกรมได้บ้าง เช่น ต้องการให้เข้าถึงตัวแปรได้เฉพาะในโปรแกรมย่อยเท่านั้น หรือให้ทุกโปรแกรมย่อยให้คลาสนั้นเห็นทั้งหมด เป็นต้น

เมื่อประกาศตัวแปรโดยใช้คีย์เวิร์ด Public นำหน้าชื่อตัวแปร ความหมายคือ ทุกส่วนของโปรแกรมสามารถมองเห็นและเข้าถึงใช้งานตัวแปรนี้ได้ ส่วนการใช้คีย์เวิร์ด Private นำหน้าชื่อตัวแปร ความหมาย คือ มองเห็นและเข้าถึงตัวแปรได้เฉพาะในโปรแกรมย่อยส่วนนั้น ดังภาพที่ 3.7



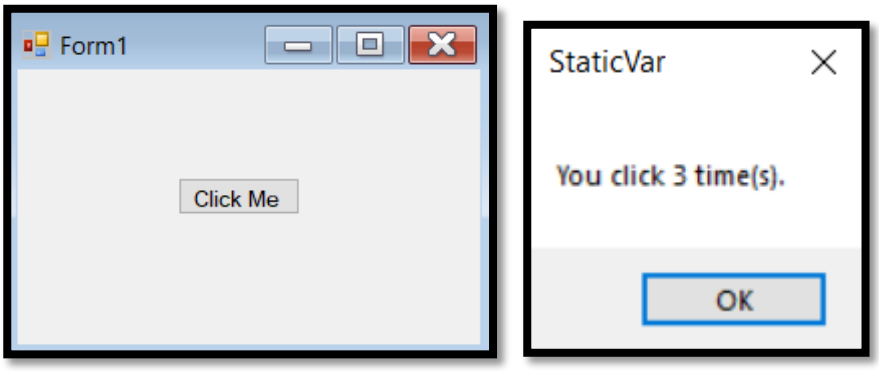

**ภาพที่ 3.7**

3.3.3 ตัวแปรแบบสเตติก (Static Variables)

ตัวแปรระดับโปรแกรมย่อย (ตัวแปรที่ถูกประกาศในโปรแกรมย่อย) จะใช้ได้เฉพาะในขณะที่การทำงานอยู่ในโปรแกรมย่อยเท่านั้น ซึ่งหลังจากโปรแกรมย่อยทำงานเสร็จแล้ว ตัวแปรนั้นก็จะถูกทำลาย และเมื่อมีการเรียกใช้งานโปรแกรมย่อยนั้นอีกครั้ง ตัวแปรระดับโปรแกรมย่อยก็จะถูกสร้างขึ้นมาใหม่ ซึ่งอาจทำให้ค่าของตัวแปรไม่ใช่ค่าเก่าอีกต่อไป แต่เราสามารถใช้คำสั่ง Static แทน Dim ในการประกาศตัวแปรเพื่อรักษาค่า

ล่าสุดของตัวแปร หลังจากเราออกจากโปรแกรมย่อยไว้ได้ โดยเมื่อเราเรียกใช้โปรแกรมย่อยอีกครั้ง ตัวแปรนี้ก็ยังคงเก็บค่าล่าสุดที่กำหนดไว้

ตัวอย่างโปรแกรม การนับจำนวนครั้งของการคลิกเมาส์ว่าปุ่มคำสั่ง จะแสดงกล่องข้อความว่าคลิกไปแล้วกี่ครั้ง




```
Public Class Form1
    0 references
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Static ClickCount As Integer
        ClickCount = ClickCount + 1
        MsgBox("You click " & ClickCount & " time(s).", MsgBoxStyle.OkOnly)
    End Sub
End Class
```

**ภาพที่ 3.8**

3.3.4 ค่าคงที่ (Constants)

ค่าคงที่มีไว้สำหรับเก็บค่าที่ไม่เปลี่ยนแปลงตลอดช่วงเวลาที่โปรแกรมรันอยู่ ซึ่งถ้าในโปรแกรมมีการใช้คำสั่งที่เปลี่ยนค่าคงที่ VB ก็จะแสดงข้อผิดพลาดขึ้นมา เพราะเราเปลี่ยนค่าที่ประกาศเป็นค่าคงที่ไม่ได้

การประกาศค่าคงที่มีลักษณะเดียวกับการประกาศตัวแปร โดยใช้คำสั่งที่มีรูปแบบดังนี้

```
[Private | Public] Const <ชื่อค่าคงที่> [As Type] = <ค่าที่กำหนดให้>
```

ในการประกาศค่าคงที่ เราสามารถกำหนดชนิดข้อมูลของค่าคงที่และกำหนดค่าเป็นตัวเลข ข้อความ ค่าในตัวแปร หรือใช้ตัวดำเนินการร่วมในการกำหนดค่าได้

ตัวอย่างการประกาศค่าคงที่

```
Const PI = 3.1415926                              'ประกาศค่าคงที่ PI
Const RPI = 1 / PI                                'ประกาศค่าคงที่ส่วนกลับของ PI
Public Const CVersion As String = "1.0.1"         'ประกาศค่าคงที่ชนิดข้อความ
Private Const OneHundred As Integer = 100         'ประกาศค่าคงที่ชนิดตัวเลข
```

และเราสามารถนำค่าที่ประกาศมาใช้ในโปรแกรมได้ดังนี้

```
AreaCircle = PI * 7 * 7
TextBox2.Text = AreaCircle
```

จากคำสั่งข้างต้น จะเป็นการหาพื้นที่วงกลมจากสูตร PI* (ความยาวรัศมียกกำลังสอง) สำหรับของเขตการประกาศค่าคงที่ จะเหมือนกับของเขตการประกาศตัวแปรดังที่ได้อธิบายไว้แล้ว

3.3.5 ชนิดของข้อมูล (Data Types)

ชนิดของข้อมูลที่เรากำหนดให้กับตัวแปรจะทำให้การเก็บข้อมูลของเรามีประสิทธิภาพ เพราะใช้พื้นที่ขนาดเหมาะสมในการเก็บข้อมูล โดยใน VB จะมีชนิดข้อมูลต่างๆ ดังนี้

| ชนิด | ขนาด (ไบต์) | คำอธิบาย |
|---|---|---|
| **ชนิดข้อมูลแบบตัวเลขจำนวนเต็ม** จะใช้เก็บเฉพาะเลขจำนวนเต็มที่ใช้บ่อยในการเขียนโปรแกรม เช่น 101, 0, -30 เป็นต้น | | |
| Byte | 1 | มีค่าตั้งแต่ 0-255 แต่ละบิตใช้แทนเลข 0 และ 1 ซึ่งเป็นเลขฐานสอง |
| Char | 2 | มีค่าตั้งแต่ 0-65535 |
| Short | 2 | มีค่าตั้งแต่ –32,768 ถึง 36,768 |
| Integer | 4 | มีค่าตั้งแต่ -2,147,483,648 ถึง 2,147,483,648 |
| Long | 8 | มีค่าตั้งแต่ –9,223,372,036,854,775,808 ถึง 9,223,372,036,854,775,808 |
| **ชนิดข้อมูลแบบเลขทศนิยม** จะใช้เก็บเฉพาะเลขจำนวนทศนิยมสำหรับใช้ในการคำนวณทางคณิตศาสตร์และวิทยาศาสตร์ เช่น 101.34, 0.22, -55.36 เป็นต้น (ตัวอักษร E จะหมายถึงยกกำลังเช่น 1.7E35 เท่ากับ $1.7 \times 10^{35}$ เป็นต้น | | |
| Single | 4 | เก็บเลขทศนิยมที่มีความละเอียดต่ำ มีค่าตั้งแต่<br>ค่าลบ -3.4028235E+38 ถึง -1.401298E-45<br>ค่าบวก 1.401298E-45 ถึง 3.4028235E+38 |
| Double | 8 | เก็บเลขทศนิยมความละเอียดสูง มีค่าตั้งแต่<br>ค่าบวก 4.94065645841246544E-324 ถึง<br>1.79769313486231570E+308 |

| ชนิด | ขนาด (ไบต์) | คำอธิบาย |
|---|---|---|
| | | ค่าลบ -1.79769313486231570E+308 ถึง<br>-4.94065645841246544E-324 และ 0 |
| Decimal | 16 | เก็บตัวเลขขนาดใหญ่มาก |
| **ชนิดข้อมูลทั่วไป** ชนิดข้อมูลทั่วไปนี้จะใช้เก็บข้อความ เก็บค่า True (จริง) หรือ False (เท็จ) อย่างในอย่างหนึ่ง และเก็บค่าวันที่ | | |
| Boolean | - | มีค่า True กับ False (ใช้ขนาดพื้นที่หน่วยความจำต่างกันขึ้นกับแพลตฟอร์ม) |
| String | - | เก็บตัวอักษรได้ 0 ถึง สองพันล้านอักขระ (2^31 และขนาดขึ้นอยู่กับแพลตฟอร์ม) |
| Date | 8 | เก็บข้อมูลที่ใช้แทนวันที่และเวลา ตัวอย่างการใช้งาน เช่น<br>`Dim MyDate As Date`<br>`MyDate = #9/05/2019#`<br>`MyDate = #2019/5/9 13:45#`<br>`MyDate = #9/05/2019 6:60AM#` |
| Object | 4/8 | 4 ไบต์บนแพลตฟอร์ม 32 บิตหรือ 8 ไบต์บนแพลตฟอร์ม 64 บิต เก็บข้อมูลที่ใช้ในการอ้างอิงถึงออบเจ็กต์ต่างๆ ถ้าเราประกาศตัวแปรโดยที่ไม่ได้ระบุชนิดข้อมูล ชนิดข้อมูลของตัวแปรตังนั้นก็จะถูกกำหนดเป็น Object ทันที เช่น Dim Test<br>ตัวแปร Test จะเป็นชนิด Object ทันที |

3.3.6 ตัวดำเนินการในการคำนวณทางคณิตศาสตร์

| การกระทำ | สัญลักษณ์ |
|---|---|
| การบวก | + |
| การลบ | - |
| การคูณ | * |
| การหาร | / |
| การหารแบบจำนวนเต็ม | \ |
| การหารเอาเศษ | Mod |
| การยกกำลัง | ^ |

3.3.7 ตัวดำเนินการในทางตรรกะ จะให้ผลลัพธ์เป็นค่า True และ False

| A | B | A And B |
|---|---|---|
| True | True | True |
| True | False | False |
| False | True | False |

| | | |
|---|---|---|
| False | False | False |

| A | B | A OR B |
|---|---|---|
| True | True | True |
| True | False | True |
| False | True | True |
| False | False | False |

| A | B | A XOR B |
|---|---|---|
| True | True | False |
| True | False | True |
| False | True | True |
| False | False | False |

| A | B | A AndAlso B |
|---|---|---|
| True | True | True |
| True | False | False |
| False | ไม่มีการหาค่า | False |

ตัวดำเนินการ **AndAlso** จะคล้ายกับตัวดำเนินการ And ยกเว้นแต่ว่าถ้านิพจน์แรกมีค่าเป็น False ก็จะไม่หาค่านิพจน์ที่สองเลย และจะให้ผลลัพธ์กับมาเป็น False

| A | B | A ORElse B |
|---|---|---|
| True | ไม่มีการหาค่า | True |
| True | False | True |
| False | False | False |

ตัวดำเนินการ **ORElse** จะคล้ายกับตัวดำเนินการ Or ยกเว้นแต่ว่านิพจน์แรกมีค่าเป็น True ก็จะไม่หาค่านิพจน์ที่สองเลย และจะให้ผลลัพธ์กลับมาเป็น True

| A | Not A |
|---|---|
| True | False |
| False | True |

3.3.8 ตัวดำเนินการในการทำงานเกี่ยวกับข้อมูลชนิด String

ตัวดำเนินการกลุ่มนี้จะใช้เชื่อม String กับ String เข้าด้วยกัน หรือ String กับข้อมูลตัวเลขก็ได้

- ใช้ + เชื่อม String กับ String เช่น

```
Console.WriteLine("Your Friend is " + YF)
```

- ใช้ & เชื่อม String กับข้อมูลแบบ Numeric หรือ String ก็ได้ เช่น

```
Console.WriteLine("Area is " & AreaCircle)
```

3.3.9 ตัวดำเนินการเปรียบเทียบ

ตัวดำเนินการประเภทนี้จะใช้เปรียบเทียบระหว่างค่า 2 ค่า โดยมีผลลัพธ์เป็น True หรือ False อย่างใดอย่างหนึ่งเท่านั้น ดังนี้

| สัญลักษณ์ของตัวดำเนินการ | ความหมาย |
|---|---|
| = | เท่ากับ |
| <> | ไม่เท่ากับ |
| < | น้อยกว่า |
| > | มากกว่า |
| <= | น้อยกว่าหรือเท่ากับ |
| >= | มากกว่าหรือเท่ากับ |

3.3.10 ลำดับในการทำงานของตัวดำเนินการ

## 3.4 รับข้อมูลจากผู้ใช้ด้วยเมธอด InputBox

เป็นช่องรับข้อมูลบนไดอะล็อกบ็อกซ์ เหมาะสำหรับการรับข้อมูลจากผู้ใช้เพียงค่าเดียว โดยกรอกข้อความหรือตัวเลขลงในช่องรับข้อมูล จากนั้นให้คลิกปุ่ม OK ข้อมูลก็จะถูกรับเข้ามาดำเนินการในโปรแกรม InputBox เป็นเมธอดของคลาส Interaction ที่สามารถอ้างอิงเรียกใช้งานได้ทั้งในโปรเจ็กต์ชนิด Windows Forms App และ Console App แต่ไม่สามารถนำไปใช้กับภาษาอื่นใน .NET ได้ เพราะเป็นเมธอดสมาชิกของเนมสเปซ Microsoft.VisualBasic ที่ไม่ใช่ .NET

<u>รูปแบบ</u>

```
'InputBox(Message,title, defaultValue, XPos,YPos)
a = InputBox("กรุณาห้อนจำนวนชั่วโมงการทำงานนอกเวลา", "คำนวณค่าแรงนอกเวลา", 0, 100, 50)
```

Message สตริงที่ต้องการให้แสดงเป็นข้อความในไดอะล็อกบ็อกซ์ เพื่อแจ้งให้ผู้ใช้ทราบว่าต้องการให้ป้อนข้อมูลอะไร มีความยาวได้ 1,024 ตัวอักษร

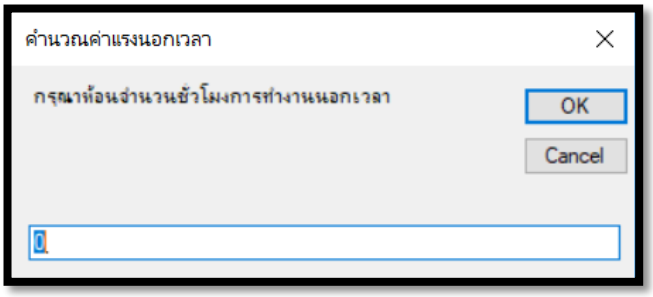
Title สตริงที่ต้องการให้แสดงเป็นข้อความในไตเติลบาร์ของไดอะล็อกซ์บ็อกซ์

defaultValue สตริงที่ต้องการให้แสดงเป็นข้อความในช่องรับข้อมูล หากผู้ใช้ไม่ป้อนข้อมูลใดๆ ลงไป ข้อความที่เป็นค่าเริ่มต้นนี้จะถูกรับเข้ามาในโปรแกรม เพื่อป้องกันไม่ให้โปรแกรมทำงานผิดพลาด

Xpos ตัวเลขระยะห่างระหว่างขอบด้านซ้ายของไดอะล็อกบ็อกซ์จากขอบด้านซ้ายของจอภาพ

Ypos ตัวเลขระยะห่างระหว่างขอบด้านบนของไดอะล็อกบ็อกซ์จากขอบด้านบนของจอภาพ

จากตัวอย่างเมื่อรันโค้ดจะได้ผลลัพธ์ดังภาพ




**ภาพที่ 3.9**

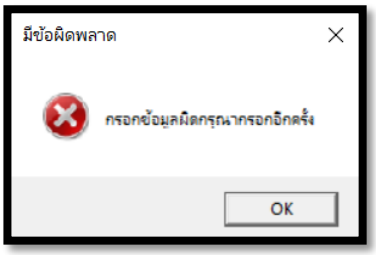
## 3.5 แสดงกล่องข้อความ MsgBox

เป็นกล่องแสดงข้อความเพื่อแจ้งผลลัพธ์จากการคำนวณ วิเคราะห์ข้อมูล แสดงสถานะการทำงาน หรือการแจ้งเตือนต่างๆ แก่ผู้ใช้ ซึ่งจะรอจนผู้ใช้รับทราบข้อมูลแล้วคลิกปุ่ม OK เพื่อยืนยันและปิด MsgBox เป็นเมธอดของคลาส Interaction ที่สามารถอ้างอิงเรียกใช้งานได้ทั้งในโปรเจ็กต์ชนิด Windows Forms App และ Console App รวมทั้งอยู่ในเนมสเปซ Microsoft.VisualBasic ที่ไม่ใช่ .NET จึงไม่สามารถรองรับการใช้งานกับภาษาอื่นใน .NET ได้

รูปแบบ

```
'MsgBox(Message, Styles, title)
MsgBox("กรอกข้อมูลผิดกรุณากรอกอีกครั้ง", MsgBoxStyle.Critical, "มีข้อผิดพลาด")
```

ผลลัพธ์



- Message สตริงที่ต้องการให้แสดงเป็นข้อความในไดอะล็อกบ็อกซ์ยาวไม่เกิน 1,024 ตัวอักษร
- Style รูปแบบของปุ่มและไอคอนที่ต้องการให้แสดงในไดอะล็อกบ็อกซ์
- Title สตริงที่ต้องการให้แสดงเป็นข้อความในไตเติลบาร์ของไดอะล็อกบ็อกซ์

รูปแบบของ Style

| **รูปแบบสไตล์** | **คำอธิบาย** |
|---|---|
| Okonly | แสดงเฉพาะปุ่ม OK |
| OKCancel | แสดงปุ่ม OK และ Cancel |
| AbortRetryIgnore | แสดงปุ่ม Abort, Retry และ Ignore |
| YesNoCancel | แสดงปุ่ม Yes, No และ Cancel |
| YesNo | แสดงปุ่ม Yes และ No |
| RetryCancel | แสดงปุ่ม Retry และ Cancel |
| Critical | แสดงไอคอนข้อความสำคัญ |
| Question | แสดงไอคอนคำถาม |
| Exclamation | แสดงไอคอนข้อความแจ้งเตือน |
| Information | แสดงไอคอนข้อความแจ้งข่าวหรือประชาสัมพันธ์ |
| DefaultButton1 | กำหนดปุ่มแรกเป็นดีฟอลต์ |
| DefaultButton2 | กำหนดปุ่มที่สองเป็นดีฟอลต์ |
| DefaultButton3 | กำหนดปุ่มที่สามเป็นดีฟอลต์ |
| ApplicationModal | กำหนดให้ผู้ใช้ตอบสนองก่อนจะกลับไปทำงานในแอพพลิเคชันต่อไป |
| SystemModal | หยุดการทำงานของแอพพลิเคชันทั้งหมดจนกว่าผู้ใช้จะตอบสนองต่อกล่องข้อความ |
| MsgBoxSetForeground | กำหนดให้กล่องข้อความปรากฏขึ้นมาซ้อนอยู่บนหน้าต่างการทำงานอื่นๆ |
| MsgBoxRight | กำหนดให้ข้อความชิดขวา |
| MsgBoxRtlReading | แสดงข้อความสำหรับการอ่านจากขวาไปซ้าย เช่น ภาษาฮิบรู และอารบิก |

ตัวอย่างการใช้เมธอด InputBox และ MsgBox

เป็นโปรแกรมคำนวณค่าแรงนอกเวลาของคนงาน โดยจะแสดงกล่องรับข้อมูลให้ผู้ใช้กรอกจำนวนชั่วโมงการทำงานนอกเวลา จากนั้นโปรแกรมจะทำการคำนวณค่าแรงด้วยชั่วโมงละ 70 บาท และแสดงค่าบนกล่องข้อความ

1) สร้างโปรเจ็กต์ใหม่ขึ้นมาชนิด Console App

2) ปรากฏหน้าต่างสำหรับเขียนโค้ด ให้เขียนดังนี้

```
Module Module1
    0 references
    Sub Main()
        Dim time, ans As Integer  'ประกาศตัวแปร time รับข้อมูล ans เก็บผลลัพธ์
        time = InputBox("กรุณาป้อนจำนวนชั่วโมงการทำงานนอกเวลา", "คำนวณค่าแรงนอกเวลา", 0, 100, 50)
        'รับข้อมูลจากกล่องรับข้อมูลเก็บในตัวแปร time
        ans = Val(time) * 70     'แปลงข้อมูลที่รับมาเป็นตัวเลขและคูรกับอัตราค่าแรง 70 บาท
        MsgBox("เวลาทำงานนอกเวลา : " & time & " ชั่วโมง ได้ค่าตอบแทน : " & ans,
               MsgBoxStyle.OkOnly, "ค่าจ้างนอกเวลา")

    End Sub
End Module
```

**ภาพที่ 3.10**

### 3.6 แสดงกล่องข้อความ MessageBox

เป็นกล่องข้อความในลักษณะเดียวกันกับ MsgBox ต่างกันที่ MessageBox เป็นคลาสใน .NET ซึ่งสามารถรองรับการใช้งานกับภาษาอื่นๆใน .NET ได้ มีเมธอดที่นิยมใช้งานได้แก่

| เมธอด | คำอธิบาย |
|---|---|
| Show(message) | แสดงเฉพาะข้อความ |
| Show(message, title) | แสดงข้อความและชื่อไตเติล |
| Show(message, title, button) | แสดงข้อความ ชื่อไตเติล และปุ่มแบบต่างๆ |
| Show(message, title, button, icon) | แสดงข้อความ ชื่อไตเติล ปุ่มแบบต่างๆ และ ไอคอนต่างๆ |

- message สตริงที่ต้องการให้แสดงเป็นข้อความในไดอะล็อกบ็อกซ์
- title สตริงที่ต้องการให้แสดงเป็นข้อความในไตเติลบาร์ของไดอะล็อกบ็อกซ์
- button กำหนดค่าแสดงปุ่มในรูปแบบต่างๆ ในไดอะล็อกบ็อกซ์

| รูปแบบปุ่ม | คำอธิบาย |
|---|---|
| AbortRetryIgnore | แสดงปุ่ม Abort, Retry และ Ignore |
| OK | แสดงปุ่ม OK |
| OKCancel | แสดงปุ่ม OK และ Cancel |

| | |
|---|---|
| RetryCancel | แสดงปุ่ม Retry แล Cancel |
| YesNo | แสดงปุ่ม Yes และ No |
| YesNoCancel | แสดงปุ่ม Yes, No และ Cancel |

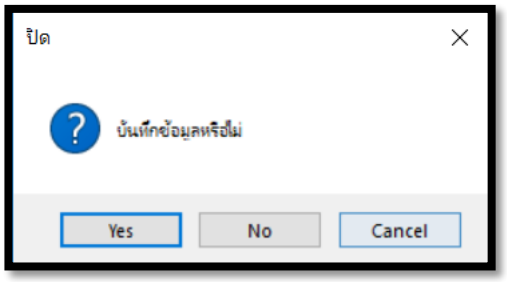
- icon กำหนดค่าแสดงไอคอนในรูปแบบต่างๆ ให้กับไดอะล็อกบ็อกซ์

| **รูปแบบไอคอน** | **คำอธิบาย** |
|---|---|
| Asterisk | แสดงไอคอนคำอธิบายประกอบ |
| Error | แสดงไอคอนแจ้งความผิดพลาด |
| Exclamation | แสดงไอคอนเครื่องหมายอัศเจรีย์ |
| Hand | แสดงไอคอนการดูแล |
| Information | แสดงไอคอนแจ้งข้อมูล |
| None | ไม่มีไอคอนสัญลักษณ์ |
| Question | แสดงไอคอนเครื่องหมายคำถาม |
| Stop | แสดงไอคอนเครื่องหมายหยุด |
| Warning | แสดงไอคอนแจ้งเตือนข้อควรระวัง |

<u>รูปแบบ</u>

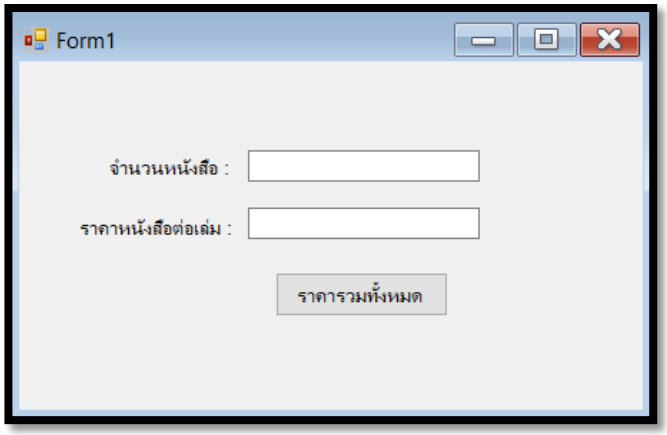
```
MessageBox.Show("บันทึกข้อมูลหรือไม่", "ปิด", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question)
```

<u>ผลลัพธ์</u>




<u>ตัวอย่าง</u> การใช้กล่องข้อความแสดงผลลัพธ์การคำนวณราคาหนังสือทั้งหมด

1) สร้างโปรเจคชนิด Windows Forms App ขึ้นมาใหม่ แล้วจัดวางคอนโทรลดังภาพ



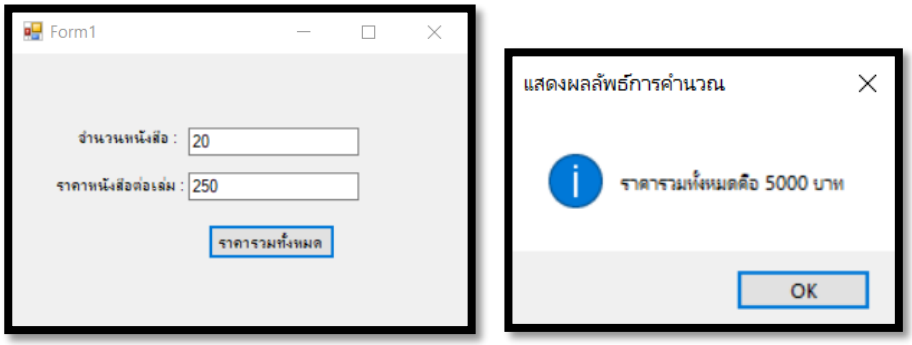

**ภาพที่ 3.11**

2) ดับเบิลคลิกปุ่มเพื่อเข้าสู่หน้าต่าง Code Editor เขียนโค้ดดังนี้

```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    Dim TotalPrice As Integer
    TotalPrice = Val(TextBox1.Text) * Val(TextBox2.Text)
    MessageBox.Show("ราคารวมทั้งหมดคือ " & TotalPrice & " บาท", "แสดงผลลัพธ์การคำนวณ",
                    MessageBoxButtons.OK, MessageBoxIcon.Information)
End Sub
```

**ภาพที่ 3.12**

3) ผลลัพธ์ที่ได้จากการรันโปรแกรม




**ภาพที่ 3.13**

## 3.7 การแปลงข้อมูลขั้นพื้นฐาน

3.7.1 ฟังก์ชั่นการแปลงข้อมูลแบบ In Line เป็นการแปลงข้อมูลโดยคอมไพล์แบบอินไลน์ที่ทำงานอย่างรวดเร็ว มีฟังก์ชันที่นิยมใช้งาน ดังนี้

| ฟังก์ชัน | ไปเป็นข้อมูลชนิด | ตัวอย่างการใช้งาน | ผลลัพธ์ |
|---|---|---|---|

| | | | |
|---|---|---|---|
| CByte | Byte | CByte("234.123") | 234 |
| CInt | Integer | CInt("3000.457") | 3000 |
| CLng | Long | CLng("10000000") | 10000000 |
| CCur | Currency | CCur("4598.8976") | 4598.8976 |
| Csng | Single | CSng("10000.7117") | 10000.71 |
| CDbl | Double | CDbl("10000.7117") | 10000.7117 |
| CDate | Date | CDate("7 Jan 2561") | 1/7/2561 |
| CStr | String | CStr(975.7773) | "975.7773" |

3.7.2 เมธอดในคลาส Convert เป็นกลุ่มของเมธอดสำหรับแปลงข้อมูลจากชนิดหนึ่งไปเป็นชนิดอื่น ซึ่งมีเมธอดที่นิยมใช้งานดังนี้

| **ชื่อ** | **รายละเอียด** |
|---|---|
| ToBoolean | เปลี่ยนจากค่าที่กำหนดไปเป็นข้อมูลชนิด Boolean ที่สอดคล้องกัน |
| ToByte | เปลี่ยนจากค่าที่กำหนดไปเป็นข้อมูลชนิด Byte (8-bit unsigned integer) |
| ToChar | เปลี่ยนจากค่าที่กำหนดไปเป็นข้อมูลชนิด Char |
| ToDateTime | เปลี่ยนจากค่าที่กำหนดไปเป็นข้อมูลชนิด DateTime |
| ToDecimal | เปลี่ยนจากค่าที่กำหนดไปเป็นข้อมูลชนิด Decimal |
| ToDouble | เปลี่ยนจากค่าที่กำหนดไปเป็นข้อมูลชนิด Double |
| ToInt16 | เปลี่ยนจากค่าที่กำหนดไปเป็นข้อมูลชนิด Short (16-bit signed integer) |
| ToInt32 | เปลี่ยนจากค่าที่กำหนดไปเป็นข้อมูลชนิด Integer (32-bit signed integer) |
| ToInt64 | เปลี่ยนจากค่าที่กำหนดไปเป็นข้อมูลชนิด Long (64-bit signed integer) |
| ToSingle | เปลี่ยนจากค่าที่กำหนดไปเป็นข้อมูลชนิด Single |
| ToString | เปลี่ยนจากค่าที่กำหนดให้อยู่ในรูปแบบสตริง |

3.7.3 เมธอด TryParse เป็นการแปลงข้อมูลชนิดข้อความให้เป็นชนิดตัวเลข โดยเมื่อแปลงข้อมูลสำเร็จจะได้ค่าตามตัวเลขที่รับข้อมูลมา แต่หากไม่สำเร็จ ผลลัพธ์ที่ได้จะเป็นค่า 0 ซึ่งโปรแกรมยังคงทำงานต่อได้ มีรูปแบบดังนี้

```
Integer.TryParse("150", Price)    'แปลงเป็นตัวเลข Integer โดยตัวแปร Price = 150
Double.TryParse("12.25", Rate)    'แปลงเป็นตัวเลข Double โดยตัวแปร Rate = 12.25
Decimal.TryParse("259", Pay)      'แปลงเป็นตัวเลข Decimal โดยตัวแปร Pay = 259
Integer.TryParse("115%", Sale)    'แปลงเป็นตัวเลข Integer โดยตัวแปร Sale = 0 (อ่านค่าไม่ได้)
```

3.7.4 เมธอด Val แปลงตัวเลขที่ถูกรวมอยู่ในข้อความให้ออกมาเฉพาะค่าตัวเลข โดยต้องขึ้นต้นด้วยตัวเลข มีรูปแบบดังนี้

```
Val("475 Baht"))          'คืนค่ากลับมาเป็น 475
Val("Baht 3778"))         'คืนค่ากลับมาเป็น 0
Val(" 452 88 Baht"))      'คืนค่ากลับมาเป็น 45288
Val("120$"))              'คืนค่ากลับมาเป็น 120
Val("32,000 Baht"))       'คืนค่ากลับมาเป็น 32
Val("30%"))               'คืนค่ากลับมาเป็น 30
```